BLOCNET: A NOVEL DEEP LEARNING MODEL FOR CYBER ATTACK DETECTION

A Thesis

by

BRANDON L. BOWEN

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2021

Major Subject: Engineering Technology

ABSTRACT

Intrusion detection systems (IDS) play a critical role in cybersecurity and are used to identify malicious behaviour in network traffic. The weakness of modern approaches is that they are reactive responses reliant on having an understanding of the types of attacks the network might encounter, or policies based on assumed user behavior. This method requires someone to analyze past attacks to develop a preventive measure and to build policies around expected user behavior. Since it relies on human intervention, it is often slow to respond and can poorly anticipate the needs of the system. It is also often inadequate because it is susceptible to any new attack that does not fit the predefined expectations for malicious activity.

Traditional machine learning (ML) models, such as Decision Tree, Random Forest or Clustering algorithms, have been used to analyze network traffic and detect attacks that are already within a system, but can range in detection time from minutes to months, and are not a preventive measure. Deep learning (DL) models provide an alternative solution that can classify data based on high-level features extracted in near real time. This means that a deep learning IDS can operate without the domain expertise and human intervention required with traditional machine learning models. DL models also tend to outperform traditional ML in both efficiency and accuracy when dealing with large datasets.

First we reviewed several anomaly-based intrusion detection datasets, such as the widely used KDD-99 and CIC-DDoS2019, and the relevant research that had used ML and DL methods for intrusion detection. Next, multiple model configurations using Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) layers were trained on the CIC-DDoS2019 dataset in order to perform multiclass classification. The results were then used to guide future work in developing a novel DL model.

For the primary contribution of the research we then designed BLoCNet, a novel IDS model based on the combination of a CNN and Bidirectional Long Short-Term Memory (BLSTM) deep learning methods that performs multiclass classification. Using an extensive preprocessing, training, and testing protocol, we show that BLoCNet is over 98% accurate when classifying the individual attacks in the CIC-IDS2017 dataset, 100% accurate with the IoT-23 dataset, and outperforms most of the related work with higher precision and recall of individual classes. We also demonstrate that BLoCNet is dataset independent by evaluating it with two topical datasets while maintaining a the high level of accuracy.

ACKNOWLEDGMENTS

This research was supported in part by the Air Force Research Laboratory (AFRL) and Department of Homeland Security (DHS) under award FA8750-19-C-0077.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Ana Goulart of the Department of Engineering Technology and Industrial Distribution and Dr. Anitha Chennamaneni of the Department of Computer Information Systems at Texas A&M Central Texas.

The literature review and presentation of data was conducted in part by Daisy Lin of the Department of Engineering Technology and Industrial Distribution.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University.

NOMENCLATURE

OGAPS	Office of Graduate and Professional Studies at Texas A&M University
B/CS	Bryan and College Station
TAMU	Texas A&M University
ESET	Electronic Systems Engineering Technology
CSCE	Computer Science and Computer Engineering
CSV	Comma Separated Values
IDS	Intrusion Detection System
ML	Machine Learning
DL	Deep Learning
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
BLSTM	Bidirectional Long Short-Term Memory
RNN	Recurrent Neural Network
CIC	Canadian Institute for Cybersecurity
DoS	Denial of Service
DDoS	Distributed Denial of Service
R2L	Root 2 Local
U2R	User 2 Root
PCAP	Packet Capture
UDP	User Datagram Protocol
SNMP	Simple Network Management Protocol

LDAP	Lightweight Directory Access Protocol
TFTP	A Trivial File Transfer Protocol
NTP	Network Time Protocol
SQL	Structured Query Language
MSSQL	Microsoft Structured Query Language
DNS	Domain Name Server
SSDP	Simple Service Discovery Protocol
IP	Internet Protocol
HTTP	Hypertext Transfer Protocol

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Research Motivation

The number of cyber attacks has grown considerably in recent years. From data breaches in multinational companies [4][5], to robot networks (known as botnets) based on the ever growing number of Internet of Things (IoT) devices [6], to ransomware holding small towns hostage [7][8], the frequency and type of attacks continues to grow. These attacks can cripple local governments, critical infrastructure, and hospitals by denying access to their systems, encrypting their data, and demanding a ransom in order to regain control. In April 2021, hackers gained access to the network of the Colonial Pipeline, a critical piece of US infrastructure, and shutdown the system with ransomware [9]. This led to two weeks of downtime, a mass panic in the Southeast United States, and a payout of $90 million. In September 2020, IBM X-Force reported a 500% year over year increase in the number of IoT based attacks [10].

An increasingly popular solution for preventing attacks is to use an Intrusion Detection Systems (IDS). The current proposals for the most successful IDSs use Deep Learning (DL) neural networks as the defining feature of their models [11]. Deep learning networks such as Convolutional Neural Network (CNNs) [12][13] and Long Short-Term Memory (LSTMs) [14][15] have been tested against datasets containing malicious network traffic and have been shown to be up to 98% accurate in classifying the attacks. However, the high overall accuracy is not always reflected in identifying the individual classes of attacks. Also, the relevance of many of the datasets used for testing leaves room for improvement. The research objective of this study is to determine if a novel deep learning model can improve the overall performance, both in overall accuracy and in identifying individual cyber attacks in the data.

1.2 Literature Review

This work builds upon previous research in using machine learning (ML) and deep learning models as an IDS. It also explores new configurations of model layers to increase the performance metrics and the models' ability to identify unique attacks. As the review of other research papers will show, deep learning models have been used with varying levels of success for cyber attack detection. However, most of the work is restricted by the use of older or limited datasets, implements only binary classification of attacks, or has poor performance when identifying unique attacks when performing multi-class classification.

1.2.1 Background on Deep Learning

This section will cover a brief explanation of the underlying layers and functions used in the proposed DL model, BLoCNet, as well as an overview of the machine learning libraries used during development. Deep learning can be thought of as the latest progression of using neural networks for performing machine learning tasks [11][16], as illustrated in Figure 1.1. Deep learning typically refers to a model that uses three or more layers of neural networks to perform a given task. By stacking layers the model is able to extract more information from the data and results in less human intervention compared to traditional ML algorithms.

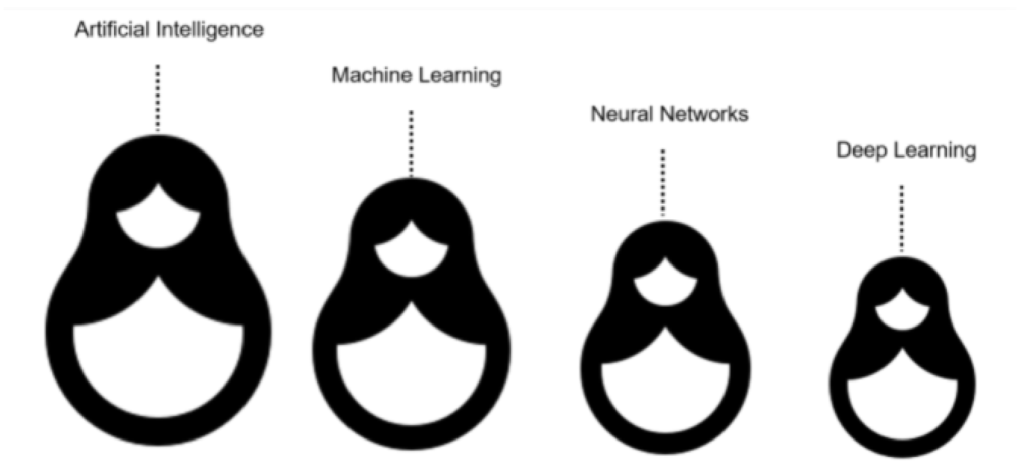


Figure 1.1: Progression of AI research as nesting dolls. [1]

1.2.1.1 ANN

An artificial neural network (ANN) is constructed from interconnected nodes, or neurons. Each node can have many inputs, where each input is assigned a different weight, but outputs only one value. They were designed as a means to replicate the neurons in the human brain, and are often visually represented in a similar manner. However, in practise they do not truly function the same way as a biological neuron. Any layer in the network that is between the input and output is considered a hidden layer and follows the same "many inputs, single output" for each neuron, as seen in Figure 1.2.

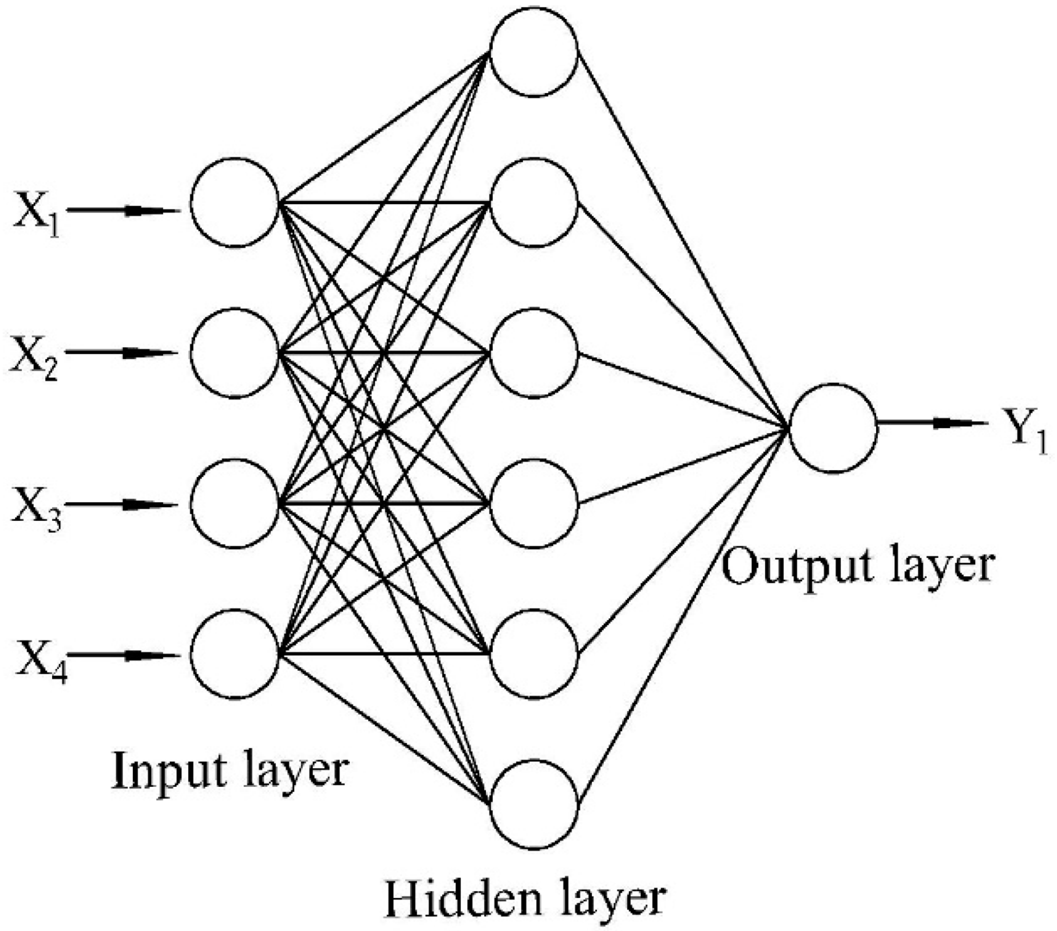


Figure 1.2: Artificial Neural Network with one hidden layer. [2]

1.2.1.2 CNN

The convolutional neural network (CNN) is one of the most widely used artificial neural network layers. CNNs are highly effective at learning local patterns within data, and the patterns themselves are translation invariant. This means that once the model recognizes a pattern, it can then be identified anywhere else in the data as being the same feature. Figure 1.3 illustrates the first layer identifying unique features, with a second layer adding context to the combinations of various features in order to recognize the image as a cat.

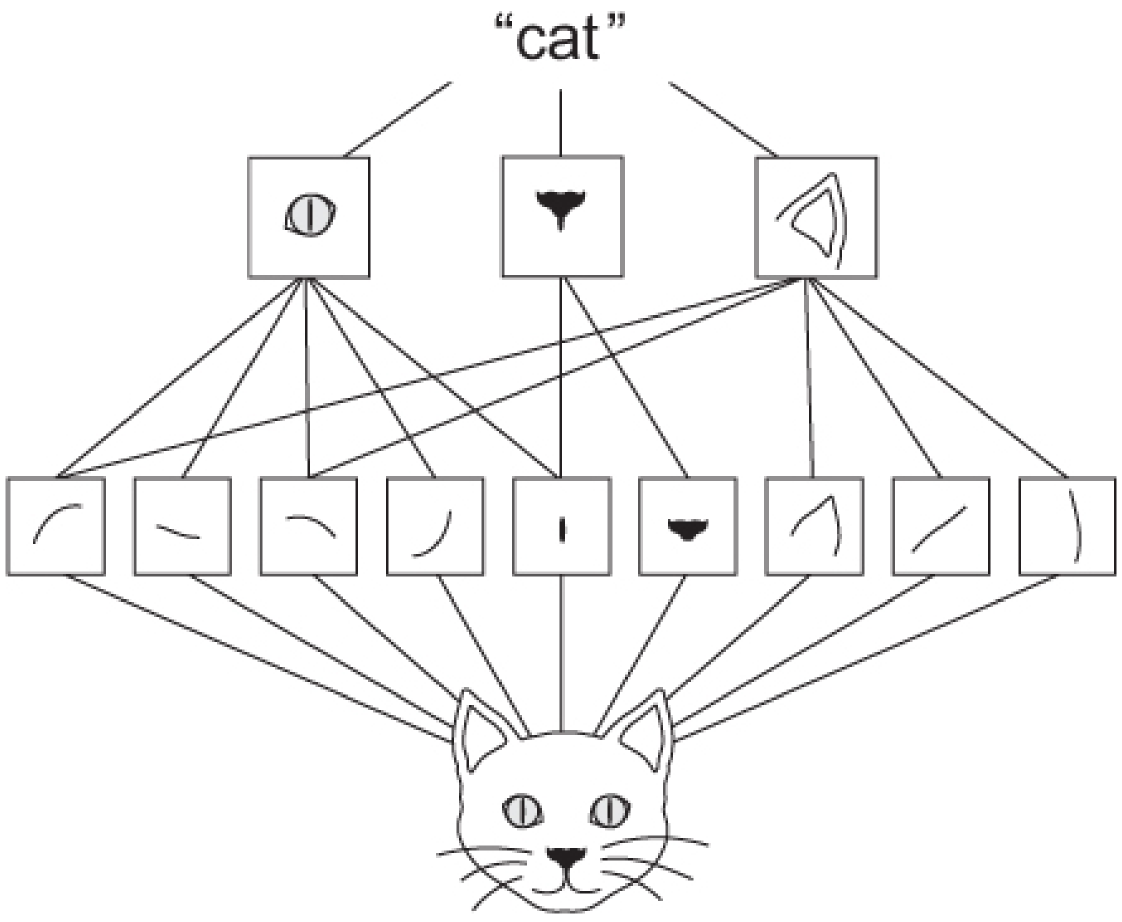


Figure 1.3: CNN feature extraction for image classification. [3]

CNNs are the current gold standard for image recognition because of their ability to recognize learned features anywhere in an image [17][18][19]. It operates by sliding a fixed-size window, or filter, across the data frame. The frame can range from a single to many dimensional tensors. CNN outputs another data frame, which is a tensor with a length equal to the number of features minus window size and added *depth* dimension consisting of the weighted features from each step of the window. A convolution is the act of applying two functions to create a third function. In the case of the CNN this is represented by applying the filter to the original input in order to create the output feature map. An example can of a 3 dimensional image with a 3x3 filter is shown in Figure 1.4.

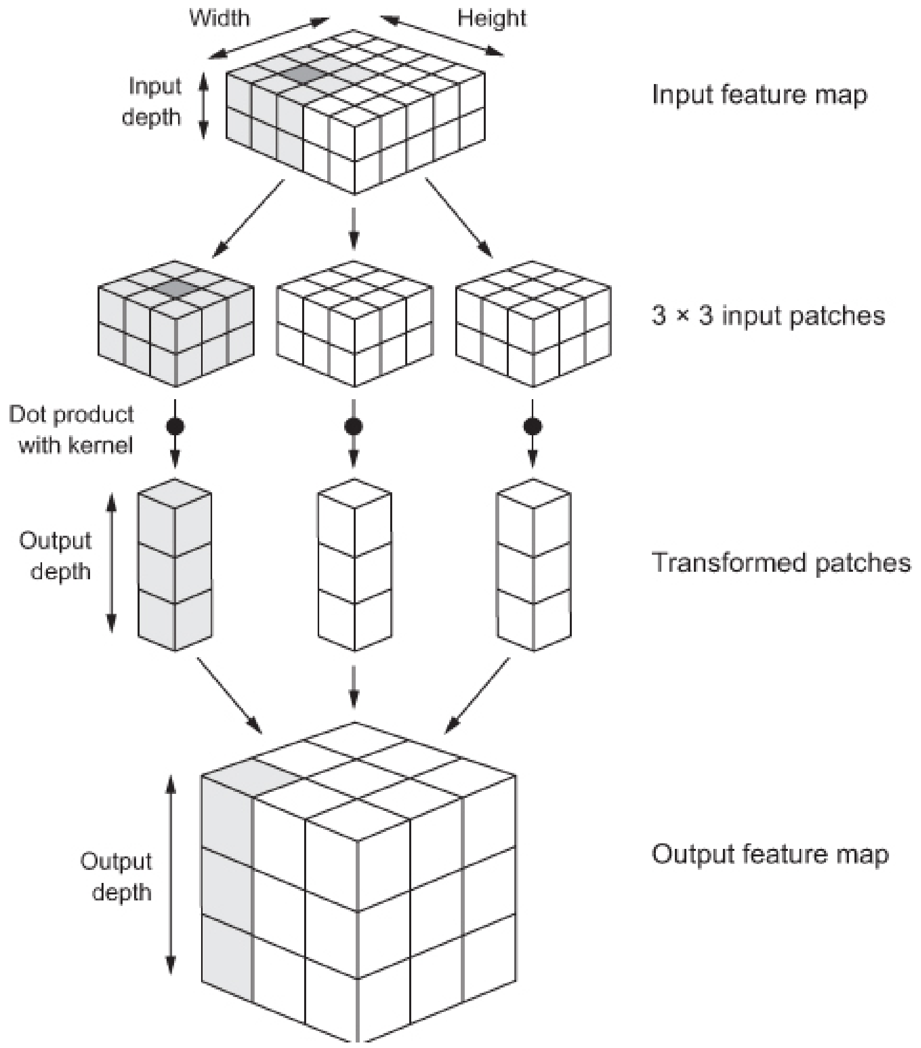


Figure 1.4: Illustration of the convolution process. [3]

1.2.1.3 RNN

A Recurrent Neural Network (RNN) uses sequential or time series data to predict the next values in the data. This is achieved by storing some of the current data in memory located in each of the neurons. The prior input data is then used to add contextual information in the next round, shown in Figure 1.5. RNNs are typically used in natural language processing (NLP), language translation, and speech recognition due to the sequential and context based nature of the data.

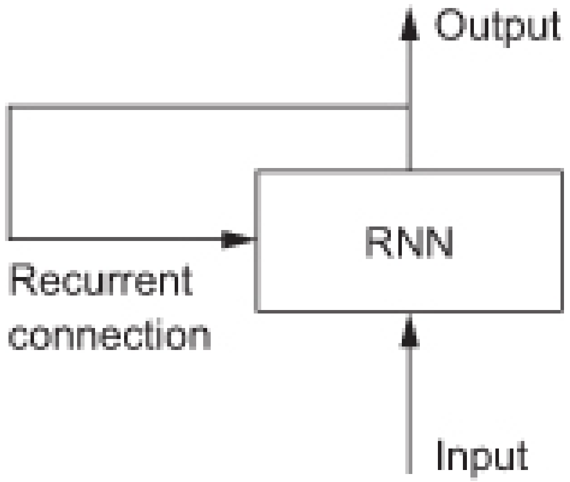


Figure 1.5: Output from the RNN is used as context of the next input. [3]

1.2.1.4 BLSTM

A Long Short-Term Memory layer is an advanced RNN layer where the previous data is carried for a longer period of time. This provides increased contextual information and results in improved prediction. A Bidirectional LSTM (BLSTM) can be thought of as two opposing LSTM layers, providing context the previous and next nodes. This was a breakthrough for speech prediction [20]. In practise they allow for more accurate prediction based on relation to the previous and upcoming words, providing grammar contextualization.

1.2.1.5 Dense

A Dense layer is a neural network where each neuron is fully connected to each of the neurons of the previous layer. This provides each neuron with all of the available information and calculates a statistical likelihood of matching each neuron's expected output. They are often used as the final, classifying layer of a model where each neuron is associated with a label or class in the dataset.

1.2.1.6 MaxPooling

Max Pooling is an operation used to quickly down sample a feature map, similar to the filter in a CNN but without as much processing time and less finesse in feature selection. Instead of applying weighted values across a filter it simply takes the maximum value within each window. It is often used in conjunction with a CNN layer to further highlight the most important features in the feature map.

1.2.1.7 BatchNormalization

Batch normalization is the process of normalizing the data between layers in the network to maintain the learned weights of in each of the layers. This is common for deep networks where the data has many opportunities for different representation at the deeper layers. By normalizing the values throughout the learning process it helps stabilize the neuron weights and can reduce the overall training time.

1.2.1.8 Dropout

A Dropout layer is a means of forcing the previous layer to forget, or drop, a percentage of random weights in the neurons. This helps prevent overfitting to the training data by forcing the previous layer to highlight more important features.

1.2.1.9 Epoch

An epoch refers to a period of time, and in the case of machine learning is the term that denotes the number of training rounds for a model.

1.2.1.10 TensorFlow

TensorFlow is an open-source software library for machine learning and artificial intelligence, with particular focus on deep neural networks. It is developed by Google and was first released in September 2015. It has sense become one of the two dominate libraries used in production and research deep learning applications. It is available for C++ and Python, with this research strictly using the Python distribution.

1.2.1.11 Keras

Keras is an open-source software library that provides Python a high level application programming interface for TensorFlow. It was developed by Francois Chollet, Senior Staff Software Engineer at Google, and released in March 2015. Keras initially supported many machine learning backends, but is now fully integrated with TensorFlow. This was the primary tool used for developing the models in this research.

1.2.2 Related Work

Table 1.1 shows an overview of the related work that was reviewed in preparation for this project. For each work, it provides a brief description of the contributions, the algorithms or models that evaluated, the datasets used for training and evaluation, and some of the results in terms of accuracy or precision. A few of these papers that are more relevant to this work, due to the models and/or datasets that were used, are discussed in detail.

Intrusion detection systems (IDS) are the first line of defense in detecting and preventing malicious behaviour. The primary weakness is that they are reactive responses reliant on prior knowledge of attacks. Much of the modern research in developing an IDS has used classical machine learning techniques to improve detection rates. Ionut Indre [22], in their work from 2016, combined several ML techniques in their model including support-vector machines (SVM), Decision Trees, and K-Means algorithms [34]. The model had successful results for the time, with precision rates of 98.4% using a Passive-Aggressive Classifier for binary and 89.5% with Decision Tree Regression for multi-class classification against the KDD-99 dataset. Research since then has steered

Reference	**Year**	**Algorithm**	**Dataset**	**Description**	**Best Performance**
[21]	2020	RNN	CIC-DDoS2019	Proposed a model for binary classification of DDoS attacks	ACC = 99%
[22]	2016	ML	KDD-99	Proposes a model consisting of many ML techniques for binary and multiclass classification	P = 98.4%
[23]	2020	DL	KDD-99, CIFAR-10	Review of DL techniques used in cybersecurity from 2014 to 2020	N/A
[24]	2019	ML	CIC-DDoS2019	Proposes a new taxonomy and dataset consisting of various DDoS attacks	N/A
[25]	2019	DL	CIC-IDS2018, KDD-99, CIFAR-10	Review of popular DL techniques at the time	N/A
[26]	2019	DL	NSL-KDD, KDD-99, ISCX, CTU-13	Review of 75 different papers on cybersecurity threats and DL methods	N/A
[12]	2021	CNN	CIC-IDS2017	Proposes a model for multiclass classification using the CIC-IDS2017 dataset	ACC = 99.8%
[27]	2021	VNN	KDD-99, CTU-13	Proposes a model combining several DL algorithms to perform binary classification	ACC = 99.9%
[28]	2021	ANN+LSTM	MAWILab	Proposes a model for multiclass classification	ACC = 74.9%
[13]	2019	PCCN	CIC-IDS2017	Proposes a model to combat unbalanced unbalanced datasets for multiclass classification	DR = 99.9%
[29]	2018	CNN	NSL-KDD	Proposes and compares CNN model against RNN for binary and multiclass classification	ACC = 99.5%
[30]	2019	CNN	KDD-99	Proposes and compares a model for binary and multiclass classification	ACC = 99.2%
[14]	2021	ML+DL	NSL-KDD, CIC-IDS2017	Proposes a hybrid model for binary and multiclass classification	ACC = 85%
[15]	2020	CNN+LSTM	NSL-KDD	Proposes model with specified DL techniques to fully use network traffic data	ACC = 84.3%
[31]	2020	ML+DL	NSL-KDD, CIC-IDS2018	Proposes a model to combat class imbalance for multiclass classification	ACC = 97%
[32]	2018	ML+DL	NSL-KDD, KDD-99, DARPA, ADFA	Review of ML and DL techniques in network IDS	N/A
[33]	2021	CNN	Bot-IoT, MQTT-IoT-IDS2020, IoT-23	Proposes a model for binary and multiclass classification for IoT attacks	ACC = 90%

Table 1.1: Literature Review Overview

towards deep learning models that have improved performance with both older, traditional datasets and newer, more topical datasets.

Early research experimenting with DL for an IDS typically involved the use of one DL algorithm at a time [32]. Two of the most common are Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). It was thought that RNN models would be well suited for a network IDS because of the contextualization of time and sequence in the data [23]. M. Elsayed *et al* [21] showed successful results when using a RNN model combined with an auto-encoder. The model was tested with the CIC-DDoS2019 dataset and was able to achieve 99% accuracy when performing binary classification, improving on the results traditional ML methods.

S. Ho *et al* [12] proposed a CNN model built from two sets of two dimensional CNN layers, each followed by a fully connected pooling layer. The model was trained on the CIC-IDS2017 dataset and used the CICFlowMeter extracted features. The use of Conv2D layers required the addition of four padding features to the dataset and then reshaping to a 9x9 tensor. The model was able to achieve 99.8% accuracy in multiclass classification, but suffered from the same issue as others using the CIC-IDS2017 in that a few classes are extremely underrepresented and results in poor recall.

K. Wu *et al* [29] proposed a similar to model that of S. Ho *et al*, two tiers CNN followed by pooling layers, in this case the pooling layers were not fully connected though. This was trained using the NSL-KDD dataset, a slightly improved and preprocessed version of the KDD-99 dataset, and produced results comparable to other RNN models and better than traditional ML techniques. They claim CNN models are better suited to larger datasets since they require fewer parameters than RNN, saving time and computational power. Although these DL algorithms prove to have higher overall accuracy, dataset imbalance prevents them from accurately detecting underrepresented attacks when performing multiclass classification [13].

I. Ullah *et al* [33] tested three variations of CNN models designed as an IDS for Internet of Things (IoT) networks, with the differences being the use of Conv1D, Conv2D, and Conv3D layers. In their data preprocessing they implemented recursive feature elimination (RFE) to reduce the number of features to 64. This was used in an attempt to extract the most relevant features prior to training and to have a feature set size appropriate for consistent training across the three models. An additional benefit is improved training time due to feature reduction. The selected features were then applied to all of the datasets prior to testing. All three CNN models proposed performed impressively with this feature selection on multiple different datasets, including the IoT-23. Each model was tested performing binary and multiclass classification, with the Conv2D model have a slight edge over the others. However, in their data processing the chose to select additional samples of underrepresented classes from other datasets but still presented the results as the performance on a single dataset, making it difficult to have a direct comparison.

Building upon the success demonstrated with CNN and RNN models, recent work has started to explore more complex CNN and RNN architectures. The work done by Y. Zhang *et al* [13] proposes a unique solution to remedy the imbalance in many datasets by implementing a parallel cross convolutional neural network (PCCN). In this configuration one branch uses CNN and the other uses a fully convolutional network (FCN). The fully convolutional layers provide up-sampling that improves training on the underrepresented classes, but at the cost of processing time. The model produced high and relatively even detection rates across all the classes in the CIC-IDS2017 dataset with an overall accuracy of 99.9%.

The BAT model proposed by T. Su *et al* [15] uses a combination of CNN and RNN layers, specifically bidirectional long short-term memory (BLSTM) layers as the RNN. In theory, the combination the two neural networks would provide the benefits of both and extract the most information from the available features. In practice, it did not solve the issue of poor classification and recall associated with the imbalanced datasets as they noticed underrepresented classes had much lower detection rates. The model was tested on the NSL-KDD dataset, which represents an out of date, imbalanced dataset, with multiple classes extremely underrepresented.

The proposed model, BLoCNet, builds upon the concepts used in the BAT model [15] and combines CNN and BLSTM layers. The results show significantly improved performance compared to the related work in all metrics tested. The model was initially trained with the CIC-IDS2017 dataset, a more recent and diverse dataset when compared to the KDD-99 and NSL-KDD datasets used in the related works. The model was then validated by training it with the IoT-23 dataset to show that it is dataset invariant. To further improve the performance, multiple sampling and feature reduction techniques were implemented along the way.

2. PRELIMINARY RESEARCH

2.1 Advanced Networking and Security Project

As preparation for my thesis research I was able to begin learning about the deep learning methods used for intrusion detection and publicly available datasets while enrolled in CSCE 665: Advanced Networking and Security during the Spring 2021 semester. The course assigned a semester long project with the goal of implementing a unique approach to any area of cybersecurity. I chose to begin testing simple deep learning models on the CIC-DDoS2019 dataset in order to guide the direction of my future research.

2.2 Dataset Evaluation

While conducting the literature review for the project, I found that one of the most commonly used datasets in early deep learning research, and even somewhat recent works, was the Knowledge Discovery and Data Cup 1999 dataset (KDD-99) [35]. It contains two weeks of benign network traffic and five weeks of network traffic containing four attacks. The attack types consist of Denial of Service (DoS), Probe, Root to Local (R2L), and User to Root (U2R). The dataset is also preprocessed and contains the feature extraction from the Wireshark packet capture (PCAP) files. While this was the standard for intrusion detection and early machine learning models, it has many downsides. As shown in Table 2.1, it is highly imbalanced with nearly 80% of the data consisting of attacks, compared to typical network traffic that would be closer to 99% benign traffic. The R2L and U2R attacks are extremely under-represented and statistically insignificant when compared to the overall size of the dataset. The dataset is also over 20 years old and not representative of modern network traffic feature extraction tools or modern attacks. Thus, it is not a quality dataset for a modern intrusion detection system.

The CIC-DDoS2019 dataset was ultimately selected for this project based the comprehensive variety and up to date attacks that it contains. It is provided by the Canadian Institute for Cybersecurity (CIC) [36]. The CIC is based at the University of New Brunswick and has become one of

Label	**Samples**	**(%)**
Normal	1033374	19.8%
DoS	4114845	78.9%
Probe	45268	0.86%
R2L	15676	0.3%
U2R	297	0.006%
Total	**5209460**	**100%**

Table 2.1: KDD99 Data Distribution

leading sources for collecting and creating both real world and simulated datasets that are intended for research and are publicly available. The data was recorded over two days and contains the simulated benign network traffic of 25 users along with the Distribute Denial of Service (DDoS) attacks. In total 12 distinct types of DDoS attacks were launched on the network:

1. UDP: A User Datagram Protocol (UDP) flood is a type of denial-of-service attack in which a large number of UDP packets are sent to a targeted server with to overwhelm that devices ability to process and respond. The firewall protecting the targeted server can also become exhausted as a result of UDP flooding, resulting in denial-of-service to legitimate traffic. A UDP flood works primarily by exploiting the steps that a server takes when it responds to a UDP packet sent to one of its ports.

2. SNMP: Simple Network Management Protocol (SNMP) reflection involves creating a flood of responses to a single spoofed IP address. During the attack, the perpetrator sends out a large number of SNMP queries with a fake IP address to multiple connected devices that reply to the fake IP address. The attack volume grows as more and more devices continue to reply, and eventually the target network is brought down under the collective volume of these SNMP responses.

3. NetBIOS: The primary purpose of NetBIOS is to allow applications on separate computers to communicate and establish sessions to access shared resources and to find each other over a local area network. The NetBIOS reflection DDoS attack generates 2.56 to 3.85 times

more response traffic sent to the target than the initial queries sent by the attacker.

4. LDAP: The Lightweight Directory Access Protocol (LDAP) is a widely used protocol for user authentication and directory services on corporate/commercial networks. By sending requests using a spoofed source IP address, that of the victim, to various servers on the internet the attacker is able to direct the server responses to the spoofed address instead of the real sender. The use of the LDAP protocol is detrimental to the victim due to the amplification of traffic: small queries made by the attacker can cause big responses from Internet servers, which in turn flood the victim with lots of packets.

5. TFTP: A Trivial File Transfer Protocol (TFTP) attack borrows from the code used in other UDP-based reflection attacks. It relies on being on able to overload the target IP address based on TFTP requests and can exceed data rates of 1 Gbps.

6. NTP: A Network Time Protocol (NTP) amplification attack is a reflection-based, volumetric distributed DDoS attack in which an attacker exploits an NTP server functionality in order to overwhelm a server with an increased amount of UDP traffic, leaving the target infrastructure inaccessible to regular queries to get the network's time information.

7. SYN: A synchronize (SYN) flood attack aims to make a server unavailable to legitimate traffic by consuming all available server resources. By flooding the server with connection request packets (TCP packet with the the SYN flag set to one), the attacker is able to overwhelm the available resources on a targeted server causing it to respond to legitimate traffic sluggishly or not at all.

8. WebDDoS: A specific DDoS attack targeting web servers by attempting to disrupt the normal traffic of the targeted server. It overwhelms the target or its surrounding infrastructure with a flood of Internet Hypertext Transfer Protocol (HTTP) traffic.

9. MSSQL: This attack occurs when a Microsoft Structure Query Language (MSSQL) Server responds to a client query or request, attempting to exploit the Microsoft SQL Server Resolu-

tion Protocol. The attack can use the protocol to leverage SQL servers by executing scripted requests using a forged IP address to make it appear that it is coming from the target server. The number of existing instances present on the affected SQL server determines the power or amplification factor of the attack.

10. UDP-Lag: This is an attack that disrupts the connection between the client and the server. This attack is often used in online gaming where the players want to slow down or interrupt the movement of other players to outmaneuver them.

11. DNS: A Domain Name Server (DNS) flood is an attack that floods a particular domains DNS servers in an attempt to disrupt DNS resolution for that domain. By disrupting DNS resolution, a DNS flood attack will compromise a website, API, or web application's ability respond to legitimate traffic. DNS flood attacks can be difficult to distinguish from normal heavy traffic because the large volume of traffic often comes from a multitude of unique locations, querying for real records on the domain, mimicking legitimate traffic.

12. SSDP: A Simple Service Discovery Protocol (SSDP) attack is a reflection-based distributed denial-of-service (DDoS) attack that exploits Universal Plug and Play (UPnP) networking protocols in order to send an amplified amount of traffic to a targeted victim, overwhelming the targets infrastructure and taking their web resources offline.

2.3 Data Analysis and Preprocessing

The CICFlowMeter [37] is a tool created by the CIC for network traffic flow generation and analysis. It takes PCAP files for input, determines the flow direction based on the first packet, and proceeds to calculate more than 80 statistical features. The features include:

'Flow ID', 'Src IP', 'Src Port', 'Dst IP', 'Dst Port', 'Protocol', 'Timestamp', 'Flow Duration', 'Total Fwd Packet', 'Total Bwd packets', 'Total Length of Fwd Packet', 'Total Length of Bwd Packet', 'Fwd Packet Length Max', 'Fwd Packet Length Min', 'Fwd Packet Length Mean', 'Fwd Packet Length Std', 'Bwd Packet Length Max', 'Bwd Packet Length Min', 'Bwd Packet Length

Mean', 'Bwd Packet Length Std', 'Flow in Bytes/s', 'Flow in Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags', 'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length', 'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s', 'Packet Length Min', 'Packet Length Max', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count', 'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count', 'URG Flag Count', 'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio', 'Average Packet Size', 'Avg Fwd Segment Size', 'Avg Bwd Segment Size', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Bulk Rate Avg', 'Bwd Bytes/Bulk Avg', 'Bwd Packet/Bulk Avg', 'Bwd Bulk Rate Avg', 'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Subflow Bwd Packets', 'Subflow Bwd Bytes', 'FWD Init Win Bytes', 'Bwd Init Win Bytes', 'Fwd Act Data Pkts', 'Fwd Seg Size Min', 'Active Mean', 'Active Std', 'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Idle Min', 'Label'

where *Fwd* and *Bwd* indicate the forward and backward flow of traffic, *IAT* denotes values calculated relative to two packets, and *ECE* is Explicit Congestion Notification Echo.

Each of the attack types and benign traffic were processed by the CICFlowMeter and saved in individual comma separated values (CSV) files, grouped by the day of their collection, resulting in a dataset of 28.9 GBytes. Given the scope of the project and limited computational power at the time, the decision was made to under-sample the dataset and use just a snapshot for training. The final distribution is shown in Table 2.2. The benign traffic was kept as the majority class with each of the attacks accounting for just under 5% of the remaining data, with the exception of WebDDoS which had very few samples.

Before training a machine learning model, the dataset generally requires some form of preprocessing. This would include accounting for missing or infinite values, removing any unnecessary features, normalizing numeric values, and encoding non-numeric values.

Label	**Samples**	**(%)**
Benign	141861	54.2%
NetBIOS	11998	4.58%
SSDP	11996	4.58%
SNMP	11996	4.58%
UDP	11995	4.58%
MSSQL	11994	4.58%
SYN	11992	4.58%
LDAP	11991	4.58%
DNS	11990	4.58%
UDP-Lag	11854	4.53%)
NTP	11848	4.53%
WebDDoS	15	(<0.001%)
Total	**261530**	**100%**

Table 2.2: Under-sampled CIC-DDoS2019 Data Distribution

For the DDoS2019 dataset, first any samples that included missing (NaN) or infinite (inf) values in any of their data fields were removed. The removal of these samples was chosen rather than replacing the values due to the varied range of the features. For instance, replacing them with a 0 or 1 value would not have been appropriate. Next, any socket information (such as *'Src IP', 'Src Port', 'Dst IP', 'Dst Port', or 'Protocol'*) or unique identifying features were removed to prevent overfitting of the data that was not representative of an attack. This included the source IP, destination IP, source port, destination port, timestamp, flow ID, and similar HTTP fields. The "Label" column was then removed from the dataset, encoded based on an index value associated with each attack type, and saved as an independent "y" value for training. The dataset was then normalized using a MinMax scalar in order to produce values ranging from 0 to 1. This is necessary for a machine learning model to effectively perform at scale and uses the formula seen in Equation 2.1.

$$ScaledValue = \frac{x - min(x)}{max(x) - min(x)} \tag{2.1}$$

2.4 ML Models Used

In order to decrease the prototyping and testing times, this project used predefined layers with standard parameters and activation functions, and each model would perform multi-class classification, i.e., identify the different types of cyber attacks. The dataset would be evaluated using four different model configurations to compare the effectiveness and potentially gain insight into a path to pursue for my thesis research. Each of the models was trained for 50 epochs.

1. 3-Layer CNN: A three-layer model consisting of an input layer with a size equal to the feature set, a single hidden layer, and dense output layer equal to the number of labels.

2. 4-Layer CNN: This used the same configuration as before with addition of a pooling layer before the dense output layer.

3. 5-Layer CNN: An auto encoder that performed feature encoding and then decoding before the dense output layer.

4. Simple RNN: A three-layer model using the Simple RNN function, a hidden layer, and a dense output layer.

2.5 Evaluation and Results

The performance of the models was assessed using the following metrics: accuracy (A), precision (P), recall (R), and F1-score (F1). They were applied at the macro level for overall performance but also on a class by class analysis. These four metrics are calculated using four variables known as the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values. A TP represents correct classification of an attack while a TN represents a correct classification of a benign packet. On the other hand, a FP indicates the incorrect classification of a benign packet while FN represents the incorrect classification of an attack packet. Using these values, accuracy, precision, recall and F1-score can be calculated using Equations 2.2, 2.3, 2.4, and 2.5.

The accuracy of the model is the percent of correctly classified samples over the total number of samples and is calculated by Eq. 2.2.

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

The precision of the model is the percent of correctly classified positive samples over the total number of predicted positive samples and is calculated by Eq. 2.3.

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

The recall of the model is the percent of correctly classified positive samples over the total number of actually positive samples and can is calculated by Eq. 2.4.

$$R = \frac{TP}{TP + FN} \tag{2.4}$$

The F1-score of the model is another way of measuring accuracy and represents the harmonic average of precision and recall. It is calculated by Eq. 2.5.

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{2.5}$$

When testing the performance of the models we placed an emphasis on the ability to accurately identify individual attacks while performing multi-class classification and compared our results with other models and related works. The accuracy and loss were also tracked during the training process for additional insight into the effectiveness of each model.

The results for each model are shown in Table 2.3. The 3 Layer CNN was the only model to produce results that approached acceptable across all performance metrics, shown in Figure 2.1, with an accuracy close to 80%. While this was the best of the models tested it was still under performing compared to much of the related work. During the training process, 20% of the training data is reserved to test the model at the end of each epoch. The results are used to adjust the weights within the model and are tracked as model accuracy and loss over each epoch. As seen in

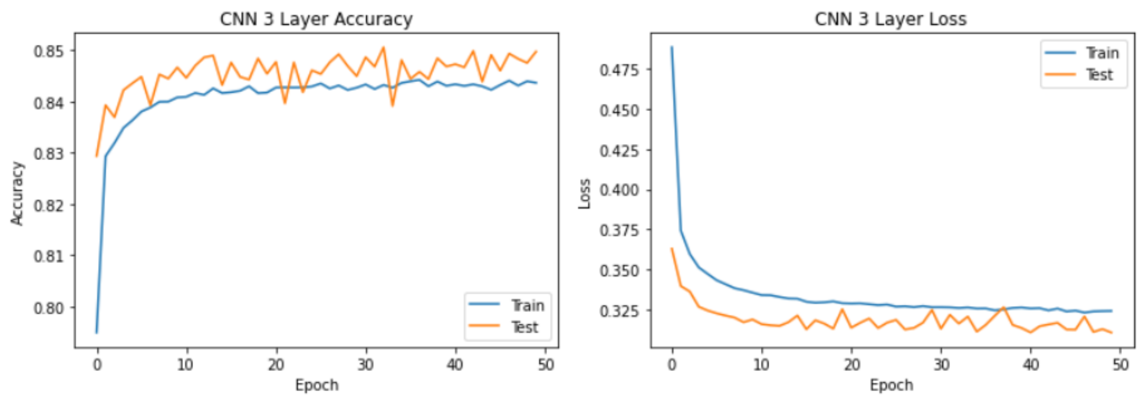


Figure 2.1: 3 Layer CNN Training Accuracy and Loss

Model	Accuracy	Loss	Precision	Recall	F1
3 Layer CNN	**79.5%**	**0.428**	**80%**	**79%**	**.76**
4 Layer CNN	59.3%	1.099	61%	63%	.58
5 Layer CNN	59%	0.985	63%	63%	.59
SimpleRNN	61.6%	0.942	55%	62%	.57

Table 2.3: Model Performance on DDoS2019

Figure 2.1, the test data following each epoch closely follows the training data. This indicates a positive training response, but the spikes are a result of the model not learning the underrepresented classes. When the training curves begin to level out it is a sign that the model may begin to over fit the data. Conversely, if the curves still showed a gradient after the 20 epochs it would suggest that the model could benefit from additional training. The other models that were tested all had very high loss and large spikes during training, resulting in low performance metrics.

2.6 Lessons Learned

This was a first step into deep learning and there were many takeaways. Under sampling the dataset was useful for quick testing and training for this project but a larger set of samples would have improved the performance of the models. The default learning rate was likely too high for the number of features in the dataset. The configuration of the model layers was also sub-optimal and needed improvement. Overall I learned a lot from this project, which prepared me for the research into novel deep learning methods.

3. BLoCNet

The primary goal of this project is to develop a novel deep learning (DL) model for cyber attack detection. After conducting the literature review and reflecting on the previous project, two unique datasets were identified for use in the training and testing of the model. The CIC-IDS2017 dataset will be used for initial prototyping and testing of the novel model, and the IoT-23 dataset as a second more topical dataset for further validation and to ensure the performance is dataset independent. The results will then be compared against the performance of other published DL models.

3.1 Dataset Evaluation

The choice of datasets was based on the following metrics for high quality data:

1. Accessibility: The data needs to be publicly available. One of the biggest challenges for IDS design is the lack of recent and relevant publicly available datasets. As data has become a valuable asset many organizations have resorted to withholding their data. With network traffic there is also a privacy issue and why many newer datasets are simulated network traffic. The CIC-IDS2017 dataset can be obtained from the CIC [36]. The IoT-23 dataset is provided by Avast [38], but can be found in preprocessed CSVs created by I. Ullah *et al* [33] at Ontario Tech University [39].

2. Completeness: The data needs to be comprehensive in its domain. Often times a dataset will contain one or more under represented classes. Depending on the relative size of the dataset this can lead to poor performance and/or require additional methods of preprocessing in order to compensate. Both datasets suffer from one or more under represented classes but have enough merit in the remaining data to justify their selection.

3. Consistency: The captured data should match the label associated with it. Both datasets were produced by reliable sources and have been scrutinized by other research.

4. Integrity: The attributes of the dataset need to be maintained. An abundance of missing entries, NaN or Inf values reduces the usefulness of the dataset. Neither dataset saw a significant reduction in samples after preprocessing.

5. Validity: The data needs to be an appropriate choice for the intended use case. Both datasets were created with the purpose of being used for IDS research.

6. Timeliness: The data should be up-to-date and relevant. The CIC-IDS2017 represents one of the newest all around IDS datasets and the IoT-23 dataset is only a year old, consisting of highly topical data in subject of IoT devices.

3.1.1 CIC-IDS2017

The CIC-IDS2017 dataset was created due to the lack of reliable datasets for anomaly based intrusion detection, with a top priority placed on diverse, realistic background traffic. The dataset contains the benign network traffic and an up to date collection of 14 common network attacks. The traffic was simulated starting Monday July 3, 2017 at 9 a.m. and ran through Friday July 7, 2017 at 5 p.m., consisting of the abstract traffic for 25 simulated users across HyperText Transfer Protocol (HTTP), HyperText Transfer Protocol Secure (HTTPS), File Transfer Protocol (FTP), Secure Shell (SSH), and email protocols, along with the directed attacks on the network. Each attack has been sorted from the network traffic packet capture (PCAP) files into their own coma separated value (CSV) files with 79 unique features being tracked using the CIC FlowMeter tool. The 14 attacks in the dataset are:

1. DoS HULK: A HTTP Unbearable Load King (HULK) attack is a DDoS attack that overwhelms the resources in a web server by flooding the server with requests for one or more URLs from multiple source machines.

2. Port Scan: A port scan is used to determine what ports on a network are open and potentially sending or receiving data. This is exploited by attackers to find vulnerabilities in network security and identify potential targets for further attacks.

3. DDoS: A Distributed Denial of Service attack. The dataset does not specify the specifics of this class.

4. DoS GoldenEye: GoldenEye is another form of HTTP flood attack buy dynamically generates the URL requests in order to randomize its agents in an attempt to keep the connection alive and avoid the detection methods used on static HTTP flood requests.

5. FTP-Patator: Patator is a multi-purpose tool used to brute force the login of numerous protocols, in this case FTP.

6. SSH-Patator: Patator brute force login on SSH.

7. DoS slowloris: Slowloris is an application layer DoS attack launched from a single computer that uses partial HTTP requests to open multiple connections and keep them open as long as possible, in effect acting as a DDoS from one machine.

8. DoS Slowhttptest: SlowHTTPTest is a tool used to simulate common, low bandwidth application layer DoS attacks.

9. Botnet: A botnet consists of many malware infected devices that are controlled remotely by a central control server. The attacker can these direct all of the devices to perform a large scale attack.

10. Web Attack - Brute Force: Brute force attacks use trial and error to guess correct login information in order to access a secure system.

11. Web Attack - XSS: Cross site scripting (XSS) is a security vulnerability that allows an attacker to cause a website to return malicious JavaScript to users.

12. Infiltration: Infiltration in cyber security is a broad term that can range from privilege escalation to a piece of malicious software on a computer. The dataset does not specify the specifics of this class.

13. Web Attack - SQL Injection: A SQL injection allows an attacker to view data that would normally not be retrievable. By submitting a database query with malicious SQL code the database can be hijacked to explore sensitive company information or private user data, and in some cases can be used to gain root access to the underlying system.

14. Heartbleed: The Heartbleed vulnerability exploits the heartbeat functionality implemented in OpenSSL, a toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. The heartbeat is a 40KB message sent from the user that is then repeated by the server to maintain the connection. However, the server does not check the size of the message received and an attacker can exploit this by sending a smaller heartbeat message. The returned message then contains the smaller heartbeat and any information in the memory until the 40KB message is filled.

The dataset contains 2,830,743 samples, with Table 3.1 showing the breakdown for each of the labels. One downside to resembling real world traffic is that the dataset is extremely unbalanced with 80% benign traffic, unequal distribution among the attacks, and many of the attacks being under represented.

3.1.2 IoT23

The IoT-23 dataset consists of network traffic from Internet of Things (IoT) devices captured from 2018 to 2019 with 20 malware captures and three benign captures [38]. The attacks were executed across multiple IoT devices as well as the benign traffic being captured from multiple devices. The PCAPs were then analyzed with the CIC-FlowMeter resulting in a large dataset of labeled attacks that matches the format of the IDS2017 dataset. The distribution of the attacks can be found in Table 3.2. This was selected as the second dataset for testing since it is very new and represents the evolving nature of cyber attacks leveraging the large quantity of IoT devices that are easily exploited. The 9 attacks in the dataset are [38]:

1. DDoS: Based on the amount of traffic flow from a device to a specific IP address that would indicate its use in a Distributed Denial of Service attack.

Label	Samples	(%)
Benign	2273097	80.3%
DoS HULK	231073	8.16%
PortScan	158930	5.65%
DDoS	128027	4.52%
DoS GoldenEye	10293	0.36%
FTP-Patator	7938	0.28%
SSH-Patator	5897	0.208%
DoS slowloris	5796	0.205%
DoS Slowhttptest	5499	0.194%
Botnet	1966	0.069%
Web Attack Brute Force	1507	0.053%
Web Attack XSS	652	0.023%
Infiltration	36	0.001%
Web Attack SQL	21	<0.001%
Heartbleed	11	<0.001%
Total	**2830743**	*100%*

Table 3.1: CIC-IDS2017 Data Distribution

2. Okiru: Indicates the connection characteristics resemble an Okiru botnet. The signature is similar to Mirai but it is a less common botnet.

3. PortScan: Port scans are used to find vulnerable targets in a network. A port scan from an IoT device indicates malicious activity because IoT devices generally use the same protocols.

4. Attack: This label indicates some type of attack from an infected device to another host, and can include brute force login, command injection in a request, etc.

5. Torii: Torii is another type of botnet and classified in the same manner as Okiru and Mirai botnets.

6. C&C: The C&C label is attached to any traffic suggesting that a device was connected to command control server. This was based on periodic connections to an unknown server or a device downloading binaries from it.

7. HeartBeat: The HeartBeat label was attached to any traffic less than 1B that had a periodic

Label	**Samples**	**(%)**
Normal	4313776	25.8%
DDoS	4619869	27.6%
Okiru	2999999	17.9%
PortScan	2999999	17.9%
Attack	1716778	10.3%
Torii	33858	0.002%
C&C	23981	0.001%
HeartBeat	12895	<0.001%
FileDownload	8035	<0.001%
Mirai	756	<0.001%
Total	**16729946**	**100%**

Table 3.2: IoT-23 Data Distribution

pattern to it, or was connected to a known C&C server.

8. FileDownload: It is rare that an IoT device would need to download any file so these events were flagged as a means to compromise the device.

9. Mirai: Mirai is an IoT malware used to create large scale botnets and carries a distinct pattern compared to other IoT botnets.

The dataset contains 16,729,946 samples, with Table 3.2 showing the breakdown for each of the labels. The dataset contains five attacks that combined makeup less than 1% of the total. However, four of them range from 4,000 to 33,000 samples and this proved to be sufficient for the model to accurately identify them. Given that Okiru and Torii are less common than Mirai, it is interesting that Okiru represents 17.9% of the dataset, Torii is 0.002%, while Mirai represents less than 0.001%.

3.2 Data Analysis and Preprocessing

3.2.1 CIC-IDS2017

In order to prevent the model from overfitting based on unique identifiers it is best practice to remove any network socket identifying information from the data. A socket is defined by the

IP address and port number. Therefore, this would include removing the source and destination IP address, source and destination port, and flow ID. The IDS2017 dataset is missing all of these to begin with other than the destination port. We chose to include this in our testing in the event that destination port alone was an identifier for any of the attack types. We chose to remove any samples that included missing (NaN) or infinite (inf) values in any of their data fields. The values in question ranged from data fields with six-digit positive or negative numbers to fields with decimal numbers out to eight-digits, suggesting that replacing with zeroes may not yield the desired results and thus chose to remove them.

The next step was to normalize the data. We used the MinMax scalar feature to normalize all values to a continuous range between zero and one. This is necessary to avoid over influence from the various units of measurement and leads to the results being more dependent on the characteristics of the data. The "Label" column was then removed from the dataset and saved as an independent "y" value for use in testing.

The initial round of testing resulted in a low average recall due to the under represented attacks and imbalance of the data overall. To correct the imbalance in the dataset we used two different oversampling methods. First, we artificially inflated samples from the five attacks that had the lowest representation of samples by multiplying the total number of entries to roughly 4,000 as a minimum. This is commonly used in data science, but it is generally not a recommended method of oversampling for ML and DL applications as it can lead to overfitting the model [40]. However, it was still used on the classes with fewer than 1,000 samples in the dataset rather than relying on chance in random sampling or grouping similar attacks into larger, more generic categories.

Next, each of the classes was split into a training and testing set using stratified sampling before being combined into the overall training and testing files [12]. This was done to ensure a balanced representation of the attacks instead of relying on the random nature of the scikit-learn (sklearn) train/test split tool. The training to testing ratio chosen was set to 0.8/0.2. In the training set, another round of oversampling was done using Synthetic Minority Oversampling Technique (SMOTE), a Python function from the imbalanced-learn (imblearn) library [40]. This method was chosen to create new attacks that were similar, but not exact duplicates of the underrepresented classes.

3.2.2 IoT-23

For initial testing the IoT-23 dataset underwent the same preprocessing steps. Despite also having imbalance issues, the number of samples for the underrepresented classes were much greater than the classes in the CIC-IDS2017. For this reason, the IoT-23 dataset performed quite well in initial testing. However, after reviewing the results of I. Ullah *et al* [33] the decision was made to revisit the preprocessing and implement the RFE algorithm with the hopes of improving overall performance. The goal was to reduce the number of features from the original 79 to just 64, with the remaining features being the most important or relevant to classification. This was implemented using the scikit-learn (sklearn) recursive feature elimination (RFE) function with a Decision Tree Classifier as the estimator. Prior to applying RFE to the data, the majority classes were significantly under sampled in order to inflate the relative presence of the underrepresented classes, resulting in 1,424,962 samples in the reduced dataset.

3.3 Model Design

The proposed novel deep learning model is BLoCNet. The BLoCNet model uses a combination of two types of deep learning neural networks: Convolutional Neural Network (CNN) and Bidirectional Long-Short Term Memory (BLSTM). This combination allows for high level feature recognition from the CNN and the contextual processing of longer sequences in the BLSTM layers [3]. A graphical outline of the model can be seen in Figure 3.1.

The data flow of BLoCNet:

1. Conv1D: The first layer of BLoCNet is a CNN layer. CNN is a deep learning algorithm that can use spatial relationships to extract and reduce features [32]. A single layer CNN has a shallow receptive field that is used for high level feature extraction from the total features in each sample. This provides a reduction of parameters and helps improve training time and performance. The layer has 64 neurons and a rectified linear (relu) activation function that

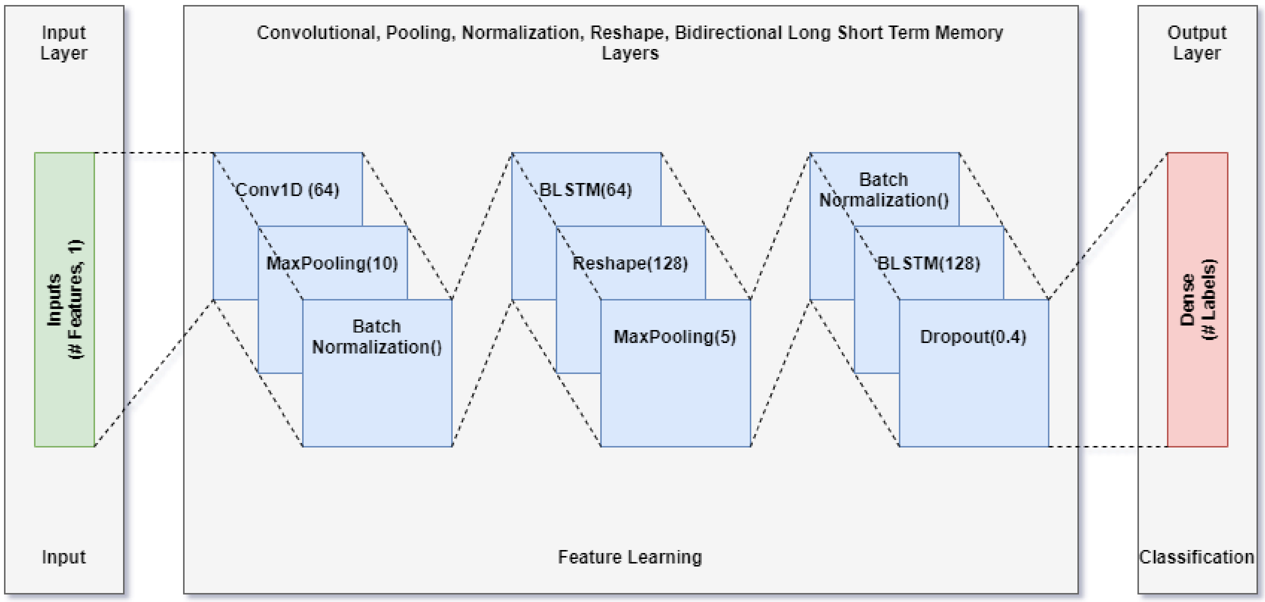


Figure 3.1: The architecture of the BLoCNet model

transforms any negative values to 0, otherwise outputs the positive value.

2. The data is then sent to a max pooling layer for additional reduction in spatial size [26]. The max pooling provides a further reduction in overall features while highlighting the most prominent features. Max pooling was used over average pooling in order to maintain the feature highlights instead of averaging the values of the feature selection. This layer has a pooling size of 10.

3. Batch normalization is then applied to the data, providing a consistent range of variables for the proceeding layer. This allows for fine tuning of the neuron weights and a faster learning rate.

4. Next is the first BLSTM layer. BLSTM can be traced back to the RNN, a recursive neural network where the output is based on the previous data [26][32]. This is useful in an IDS because many attacks have a pattern that can be identified in the sequence of features. LSTM is an improved version of RNN that utilizes memory cells and forget gates [23][32]. It also solves the vanishing gradient problem, which can occur as the input to a cell grows too

large causing errors in the network [3][41]. The BLSTM uses both forward and backwards LSTMs [15] for added contextualization. This layer has 64 units, which defines length of the memory based cells, and gives an output of with a shape of 128.

5. Reshape is then applied to the date in order to add an additional "Null" dimension to the tensor. This dimension is lost in the previous BLSTM layer but expected by the following layers.

6. Max pooling is applied one more time, with a pooling size of 5 for feature reduction.

7. Another Batch normalization is performed prior to the second BLSTM layer.

8. The data is then processed by the second BLSTM layer, this time with 128 units in order to increase the feature map prior to classification.

9. A Dropout function set to 40% is applied to the BLSTM output. This will ignore a random selection of 40% of the output features and helps in tuning the neuron weights and highlight the more important features.

10. A Dense layer is used for classification of the output. This layer contains a number of neurons equal to the labels in the dataset, with each neuron connected to all of the outputs from the second BLSTM. It uses a softmax activation to perform probabilistic classification of the data.

The model is compiled with the Adam optimizer with learning rate of 0.0001 and loss function of sparse categorical crossentropy in order to perform multiclass classification. It is then trained over 20 epochs, with a train/test split of 20%, and a validation split of 20% applied to the training data.

3.4 Evaluation

The performance of BLoCNet was also assessed using accuracy (A), precision (P), recall (R), and F1-score (F1). These were presented in Section 2 and can be calculated using Equations 2.2, 2.3, 2.4, and 2.5. will be applied at the macro level for overall performance but also on a class by class analysis. These four metrics are calculated using four variables known as the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) values. A TP represents correct classification of an attack while a TN represents a correct classification of a benign packet. On the other hand, a FP indicates the incorrect classification of an attack packet while FN represents the incorrect classification of a benign packet. For this model an emphasis is placed on the ability to accurately identify individual attacks while performing multiclass classification and the results will be compared with other models and related works.

In summary, the design of BLoCNet leverages the strength of both CNN and BLSTM layers for intrusion detection. Two datasets very identified due to their relevancy to modern attacks, recent release, and covering to unique aspects of intrusion detection. In addition, a method for preprocessing the datasets was proposed, including different techniques for handling data imbalances.

4. RESULTS

This chapter highlights the results of BLoCNet when performing multiclass classification on the CIC-IDS2017 and IoT-23 datasets. In both cases the model was trained twice; once with the original dataset after preprocessing, and again with a sampling method applied to the data.

4.1 CIC-IDS2017

The initial testing of the CIC-IDS2017 gave promising results but still had room for improvement. BLoCNet had an accuracy of 96%, precision of 96%, recall of 96%, and F1 score of .96. However, the model struggled to properly classify most of the underrepresented classes, as seen in the confusion matrix in Figure 4.1

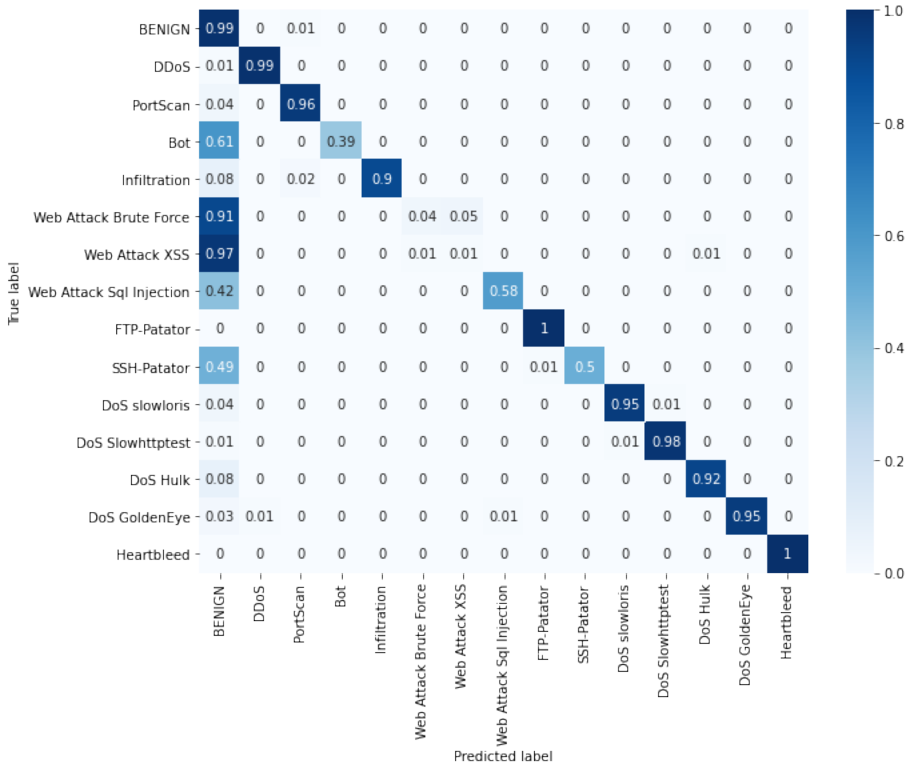


Figure 4.1: Confusion matrix for IDS2017 dataset

4.2 CIC-IDS2017 with SMOTE

The SMOTE oversampling technique was then applied to the CIC-IDS2017 dataset and tested again. The BLoCNet model saw increased performance in all metrics with an overall accuracy of 98%, with precision, recall, and F1 score of 90%, 83%, and 81, respectively. The model also scored high metrics for all of the labels in the dataset, including the under represented attacks, as seen in the confusion matrix in Figure 4.2.

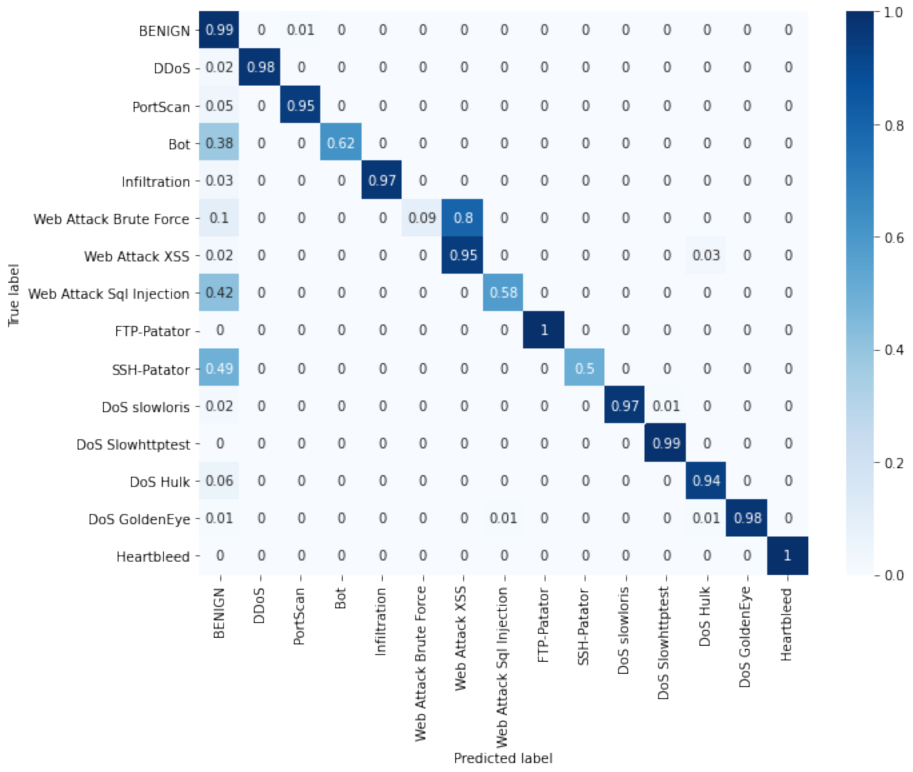
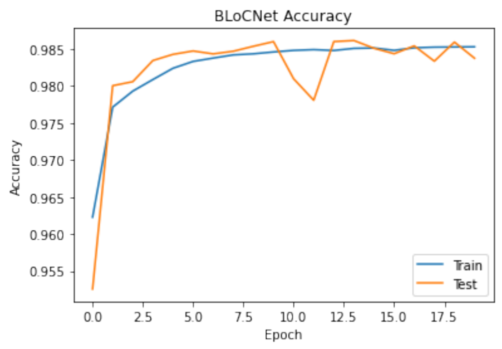
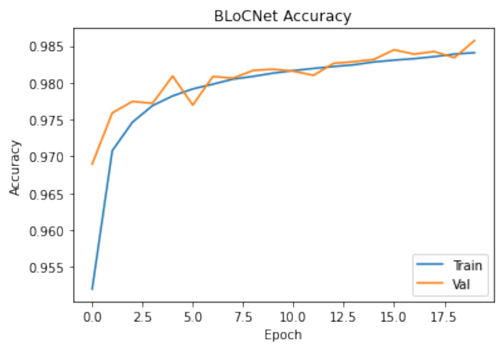


Figure 4.2: Confusion matrix for IDS2017 dataset with SMOTE

When analyzing the training accuracy and loss it is easy to see the improvement from using SMOTE. Figure 4.3b shows a much smoother curve for the test data and Figure 4.4b has a corresponding loss curve. This is to be expected with the increased samples of underrepresented classes due to SMOTE.

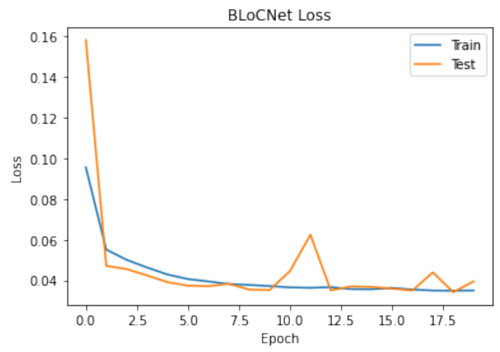


(a) CIC-IDS2017

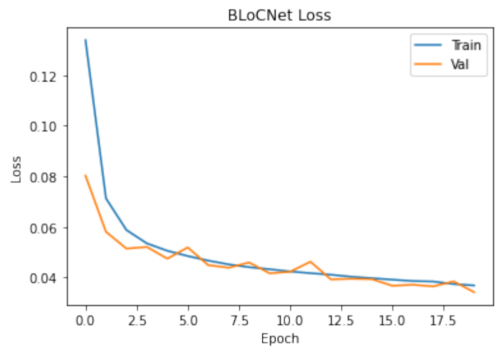


(b) CIC-IDS2017 with SMOTE

Figure 4.3: Training accuracy



(a) CIC-IDS2017



(b) CIC-IDS2017 with SMOTE

Figure 4.4: Training loss

For comparative results we also tested the SMOTE oversampled IDS2017 results against other deep learning models. The results are shown in Table 4.1. While both the single and three layer CNN models have high accuracy and precision, they lack the recall of individual attack classification. This is likely attributed to the imbalance in the dataset and the high success rate using a CNN for feature extraction. However, BLoCNet had the highest F1 score, showing a better balance of precision and recall among each of the labels.

Model	**Accuracy**	**Precision**	**Recall**	**F1**
BLoCNet	98%	90%	83%	81
CNN: 3 Layer	98%	88%	75%	77
CNN: 1 Layer	98%	93%	76%	78
RNN: LSTM	97%	74%	67%	69
Autoencoder	98%	80%	75%	76
DNN	98%	91%	73%	76

Table 4.1: Multiclass Classification Model Performance on IDS2017

4.3 IoT-23

The initial testing of BLoCNet with the IoT-23 dataset resulted in an accuracy of 100%, precision of 100%, recall of 100%, and F1 score of 1.0. However, the model struggled with two of the underrepresented classes but due to the extreme imbalance of the dataset their impact was negligible in the overall results, as seen in Figure 4.5. The initial results were viewed as satisfactory until the publication of the work by I. Ullah *et al* [33] in July.

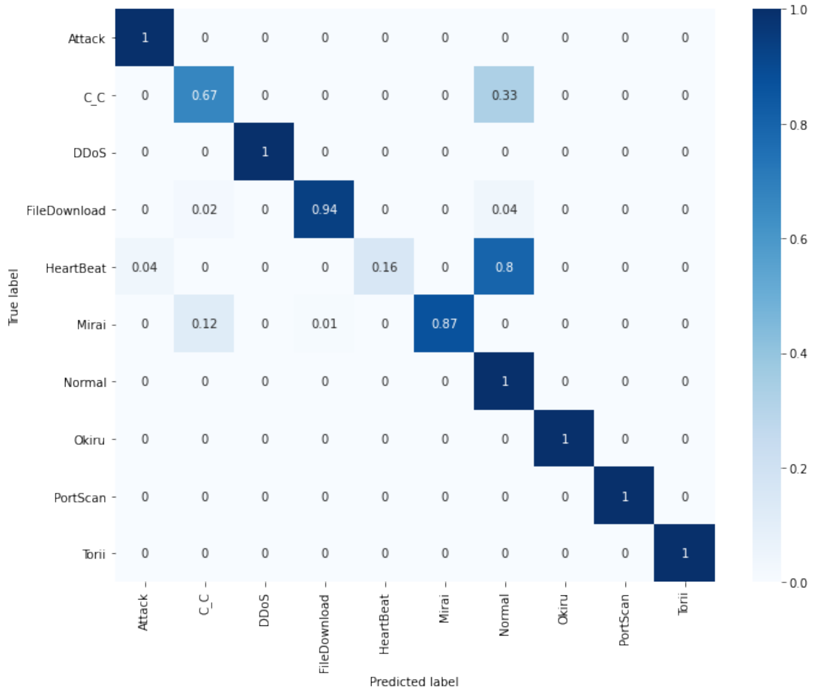


Figure 4.5: Confusion matrix for IoT-23 dataset

4.4 IoT-23 under sampled with RFE

In order to improve performance on the IoT-23 the preprocessing of the dataset borrowed the RFE application used by I. Ullah *et al* [33], as well as undersampling the majority classes. This resulted in near perfect classification for all classes, with the most underrepresented class achieving 91%. Overall, the model maintained an accuracy of 100%, precision of 100%, recall of 100%, and F1 score of 1.0. Figure 4.6 shows the final confusion matrix for testing of BLoCNet with the IoT-23 dataset.

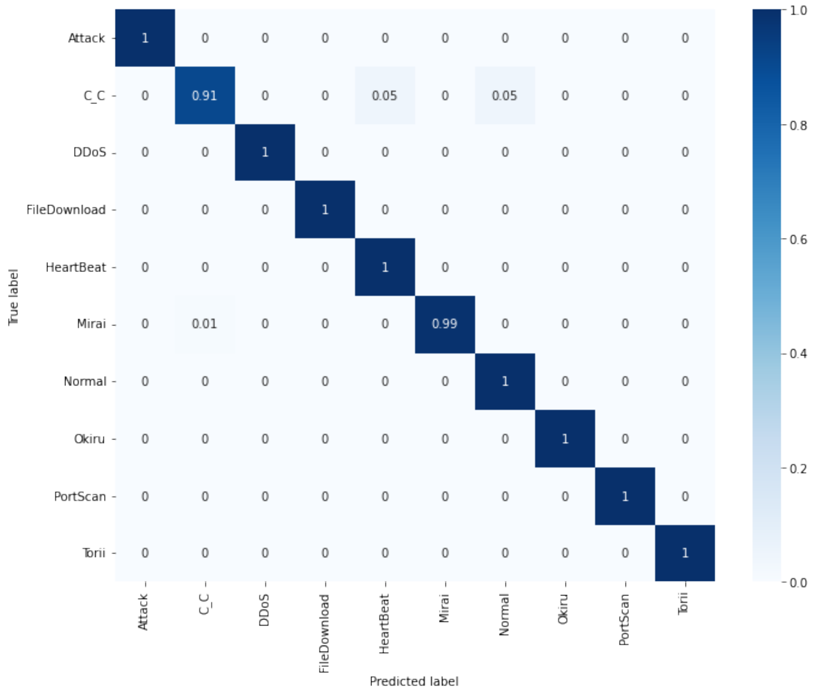
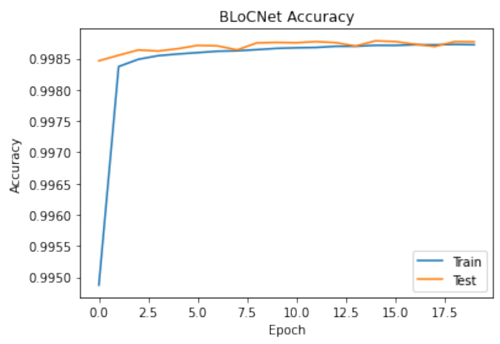
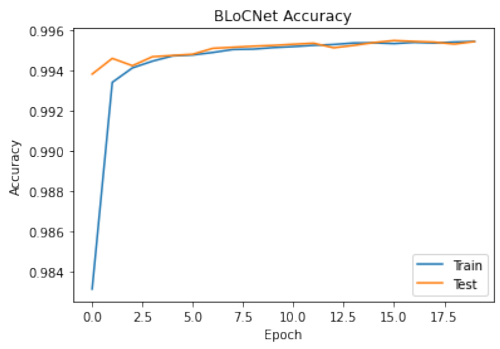


Figure 4.6: Confusion matrix for IoT-23 dataset under sampled with RFE

Again we see an improvement in the training accuracy and loss, but in this case due to undersampling and the use of RFE. Figure 4.7b shows a slightly smoother curve for the test data but Figure 4.8b has avoids the large spikes in the loss curve.

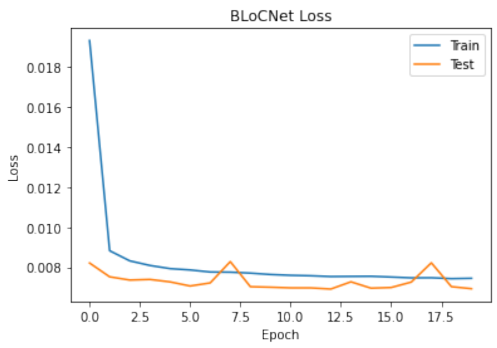


(a) IoT-23

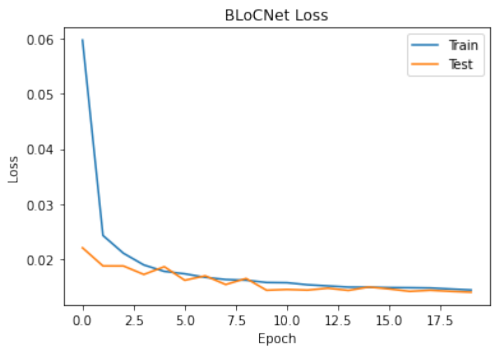


(b) IoT-23 under sampled with RFE

Figure 4.7: Training accuracy



(a) IoT-23



(b) IoT-23 under sampled with RFE

Figure 4.8: Training loss

4.5 Comparison to Related Work

Table 4.2 shows a summary of the performance metrics for BLoCNet with each of the datasets.

The BAT model [15] uses a similar network architecture but is trained on the NSL-KDD dataset, an older and less diverse dataset than the CIC-IDS2017 that we used. Their model was only able to achieve 84% accuracy when performing multiclass classification on just 5 classes.

Comparing our results with the PCCN [13] we see similar accuracy and recall across all attacks, however the PCCN model is using a unique feature extraction method that greatly reduces the number of features in the data as well reducing the number of attacks by grouping under rep-

Dataset	**Accuracy**	**Precision**	**Recall**	**F1**
CIC-IDS2017*	96%	96%	96%	.96
CIC-IDS2017 w/ SMOTE	98%	98%	98%	.98
IoT-23*	100%	100%	100%	1.0
IoT-23 under sampled w/ RFE	100%	100%	100%	1.0

Table 4.2: Summary of BLoCNet Performance
*poor classification of underrepresented classes

resented labels into general purpose labels. This could lead to overfitting by training a single label with multiple attack variations and false classification due to the reduced feature size.

When comparing the results on the IoT-23 dataset, I. Ullah *et al* [33] have shown slightly higher accuracy and recall of the underrepresented classes. However, it is difficult to make a direct comparison due to their method of borrowing data from multiple datasets. Overall, BLoCNet has a higher level of accuracy and a leading F1 score for the IoT-23 dataset compared to other reviewed works.

These results show BLoCNet to be superior to other methods in overall accuracy, but particularly when identifying each of the unique attack types in a dataset. Where other models have made compromises by using older datasets, drastically reducing the number of features, or combining underrepresented classes in order to increase overall accuracy, we were able to maintain the majority of features and accurately detect each of the unique classes in the dataset.

5. CONCLUSION

5.1 Summary

This work explores a novel IDS framework, BLoCNet, for cyber attack detection using a hybrid deep learning network composed of CNN and BLSTM layers. The convolutional layer allows the model to recognize patterns in the features of the network traffic at a high level but in a relatively fast computation time before passing into the slower BLSTM layers. The BLSTM layers then capitalize on the forward and backward propagation of data to identify the attacks within the data before classifying the sample by type of attack. BLoCNet differentiates itself from similar work in that it uses a unique network architecture that takes advantage of the benefits offered by both CNN and BLSTM layers in such a way that results in improved accuracy, precision, recall, and F1 score compared to similar work. The primary improvement is in the classification of under represented attacks in the CIC-IDS2017 and IoT-23 datasets.

5.2 Limitations

The BLoCNet model has some limitations. Many researchers have shown binary classification to be trivial with most neural network architectures [21][25][29][30][14] which has the potential to be useful as an early warning IDS but misses out on identifying the specific attack infiltrating a network. The BLoCNet model uses a multiclass approach in order to provide more useful information as an IDS with the goal of retaining the high accuracy in detection associated with binary classification. The downside to this is that we need to account for unknown attacks in some way since they may not resemble any of the known 15 attacks in the training dataset.

Another consideration is the removal of unique identifier features in the dataset and/or a reduction in the overall features. Most researchers chose to remove the source and destination IP address, port number, and timestamp in order to prevent overfitting [21][12], while others chose to implement their own feature extraction tool [13]. The CIC-IDS2017 dataset is already missing the source IP, source port, and destination IP but still keeps the destination port. We chose to include this feature in the event that it correlates with a specific type of attack although this may lead to overfitting. To verify our theory we tested the BLoCNet model with the same oversampled data but without the destination port. In this test we did not observe any improvement to overall metrics or unique attack classification. Actually we saw a reduction in performance of the model.

5.3 Future Work

In the future, the goal is to continue improving the detection rate for under represented attacks while maintaining high performance metrics. This could potentially be achieved by using additional feature reduction, different oversampling techniques, or a combination of both. Further analysis of the features created by the CICFlowMeter combined with an understanding of the attack signatures could potentially add insight to critical features. The IoT-23 dataset showed improved performance when under sampled but could also be tested with SMOTE. This would increase the underrepresented classes while maintaining all of the original samples in the majority classes. The model could also be trained on newer datasets as they become available, or a collection of topical datasets as seen with I. Ullah *et al* [33].

5.4 Contributions

This research work makes several contributions to the field of deep learning for cybersecurity intrusion detection:

1. reviewed the relevant work that applied deep learning techniques for use as intrusion detection systems, including recently published work during the conclusion of testing BLoCNet.

2. proposed a novel IDS, BLoCNet, that uses a combination of CNN and BLSTM layers. This hybrid approach leverages the high level feature extraction of the CNN along with the bidirectional contextualization of the BLSTM layers.

3. examined the performance of the model and compared it with related work. Blocnet exceeded the performance of previous models across all performance metrics that we tracked, but specifically the classification of unique attacks in the CIC-IDS2017 and IoT-23 datasets.

4. proposed three methods of preprocessing the data in order to compensate for the under represented attacks:

    - by oversampling the under represented attacks to a baseline value before incorporating a Synthetic Minority Oversampling TEchnique (SMOTE) in order to provide diversity of the oversampled values.
    - by reducing the number of features in the dataset using RFE the training time was reduced while increasing the accuracy, precision, and recall across all attacks in the datasets.
    - by under sampling a sufficiently large dataset in order to scale down the over represented classes and increase the accuracy of identifying under represented classes.

REFERENCES

[1] I. C. Education, "Ai vs. machine learning vs. deep learning vs. neural networks: What's the difference?," 27 May, 2020. https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks.

[2] H. Chai, X. Chen, Y. Cai, and J. Zhao, "Artificial neural network modeling for predicting wood moisture content in high frequency vacuum drying process," *Forests*, vol. 10, no. 1, 2019. https://www.mdpi.com/1999-4907/10/1/16.

[3] F. Chollet, *Deep Learning with Python.* Shelter Island, NY: Manning Publishing Company, 2018.

[4] R. Satter, "Up to 1,500 business affected by ransomware attack," July 2021. https://www.reuters.com/technology/hackers-demand-70-million-liberate-data-held-by-companies-hit-mass-cybe

[5] M. Hill and D. Swinhoe, "The 15 biggest data breaches of the 21st century," July 2021. https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html.

[6] D. McMillen, "Internet of threats: Iot botnets drive surge in network attacks," April 2021. https://securityintelligence.com/posts/internet-of-threats-iot-botnets-network-attacks/.

[7] C. Velazco and R. Lerman, "Shut down everything: Global ransomware attack takes a small maryland town offline," July 2021. https://www.washingtonpost.com/technology/2021/07/08/kaseya-ransomware-attack-leonardtown-maryland/.

[8] B. Allyn, "22 texas towns hit with ransomware attack in 'new front' of cyberassault," August 2019. https://www.npr.org/2019/08/20/752695554/

23-texas-towns-hit-with-ransomware-attack-in-new-front-of-cyberassault.

[9] W. Turton and K. Mehrotra, "Hackers breached colonial pipeline using compromised password," 4 June, 2021. https://www.bloomberg.com/news/articles/2021-06-04/hackers-breached-colonial-pipeline-using-compromised-password.

[10] D. McMillen, W. Gao, and C. DeBeck, "A new botnet attack just mozied into town," 17 September, 2020. https://securityintelligence.com/posts/botnet-attack-mozi-mozied-into-town/.

[11] S. Mahapatra, "Why deep learning over traditional machine learning," March 2018. https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177

[12] S. Ho, S. A. Jufout, K. Dajani, and M. Mozumdar, "A novel intrusion detection model for detecting known and innovative cyberattacks using convolutional neural network," *IEEE Open Journal of the Computer Society*, vol. 2, pp. 14–25, 12 January, 2021.

[13] Y. Zhang, X. Chen, D. Guo, M. Song, Y. Teng, and X. Wang, "Pccn: Parallel cross convolutional neural network for abnormal network traffic flows detection in multi-class imbalanced network traffic flows," *IEEE Access*, vol. 7, pp. 119904–119916, 05 August, 2019. https://doi.org/10.1109/ACCESS.2019.2933165.

[14] C. Liu, Z. Gu, and J. Wang, "A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning," *IEEE Access*, vol. 9, pp. 75729–75740, 21 May, 2021. https://doi.org/10.1109/ACCESS.2021.3082147.

[15] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "Bat: Deep learning methods on network intrusion detection using nsl-kdd dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 10 February, 2020. https://doi.org/10.1109/ACCESS.2020.2972627.

[16] I. C. Education, “Ai vs. machine learning vs. deep learning vs. neural networks: Whats the difference?,” 27 May, 2020. https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks.

[17] S. Kido, Y. Hirano, and N. Hashimoto, “Detection and classification of lung abnormalities by use of convolutional neural network (cnn) and regions with cnn features (r-cnn),” in *2018 International Workshop on Advanced Image Technology (IWAIT)*, pp. 1–4, 2018. https://doi.org/10.1109/IWAIT.2018.8369798.

[18] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, “Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, 2016. https://doi.org/10.1109/TMI.2016.2528162.

[19] J. Robin, M. R. Soni, R. R. Dubey, N. A. Datkhile, and J. Kolap, “Computer vision for hand gestures,” in *2020 International Conference on Convergence to Digital World - Quo Vadis (ICCDW)*, pp. 1–4, 2020. https://doi.org/10.1109/ICCDW45521.2020.9318682.

[20] Y. Ning, Z. Wu, R. Li, J. Jia, M. Xu, H. Meng, and L. Cai, “Learning cross-lingual knowledge with multilingual blstm for emphasis detection with limited training data,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5615–5619, 2017. https://doi.org/10.1109/ICASSP.2017.7953231.

[21] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, “Ddosnet: A deep-learning model for detecting network attacks,” pp. 391–396, 09 October, 2020. https://doi.org/10.1109/WoWMoM49955.2020.00072.

[22] I. Indre and C. Lemnaru, “Detection and prevention system against cyber attacks and botnet malware for information systems and internet of things,” pp. 175–182, 10 November, 2016. https://doi.org/10.1109/ICCP.2016.7737142.

[23] P. Dixit and S. Silakari, "Deep learning algorithms for cybersecurity applications: A technological and status review," *Computer Science Review*, vol. 39, p. 100317, 2021. https://www.sciencedirect.com/science/article/pii/S1574013720304172.

[24] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," pp. 1–8, 2019. https://ieeexplore.ieee.org/document/8888419.

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

[26] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, 2019. https://www.mdpi.com/2078-2489/10/4/122.

[27] M. H. Haghighat and J. Li, "Intrusion detection system using voting-based neural network," *Tsinghua Science and Technology*, vol. 26, no. 4, pp. 484–495, 04 January, 2021. https://doi.org/10.26599/TST.2020.9010022.

[28] M. Kim, *Early Network Attack Identification*. PhD thesis, 2021. =https://www.proquest.com/dissertations-theses/early-network-attack-identification/docview/2533142789/se-2.

[29] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50850–50859, 06 September, 2018. https://doi.org/10.1109/ACCESS.2018.2868993.

[30] R. U. Khan, X. Zhang, M. Alazab, and R. Kumar, "An improved convolutional neural network model for intrusion detection in networks," pp. 74–77, 03 October, 2019. https://doi.org/10.1109/CCC.2019.000-6.

[31] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," *IEEE Access*, vol. 9, pp. 7550–7563, 30 December, 2021. `https://doi.org/10.1109/ACCESS.2020.3048198`.

[32] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 15 May, 2018. `https://doi.org/10.1109/ACCESS.2018.2836950`.

[33] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in iot networks," *IEEE Access*, vol. 9, pp. 103906–103926, 01 July, 2021. `https://doi-org/10.1109/ACCESS.2021.3094024`.

[34] G. Rebala, A. Ravi, and S. Churiwala, *An Introducation to Machine Learning*. Cham Springer, 2019. `https://doi.org/10.1007/978-3-030-15729-6`.

[35] D. B. S. of Information and I. Computer Sciences, University of California, "Kdd cup 1999 data," 1999. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[36] U. of New Brunswick, "Intrusion detection evaluation dataset (cic-ids2017)," 2017. `https://www.unb.ca/cic/datasets/ids-2017.html`.

[37] U. of New Brunswick, "Cicflowmeter," 2017. `https://www.unb.ca/cic/research/applications.html#CICFlowMeter`.

[38] . M. J. E. Sebastian Garcia, Agustin Parmisano, "Iot-23: A labeled dataset with malicious and benign iot network traffic," 2020. `https://www.stratosphereips.org/datasets-iot23`.

[39] O. T. University, "Iot intrusion detection datasets," 2021. `https://sites.google.com/view/iotdataset1`.

[40] M. Altini, "Dealing with imbalanced data: undersampling, oversampling, and proper cross-validation," August 2015. `https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cros`

[41] K. Kim, M. Aminanto, and H. Tanuwidjaja, *Network Intrusion Detection using Deep Learning*. Spring Briefs on Cyber Security Systems and Networks, 2018.