HIGH SCHOOL EDITION
Journal of Student Research
Volume 11 Issue 3 (2022)

A Method for Network Intrusion Detection Using Deep Learning

Nihar Mudigonda

Rocklin High School, Rocklin, CA, USA

ABSTRACT

In an increasingly digitally reliant world, organizations are facing the ever more challenging problem of how to best defend their digital information and infrastructure. Current non-machine learning methods for detecting network intrusion, like signature-based and anomaly-based algorithms, are slow and unreliable. Signature based detection holds signatures, or known information and warning signs, about a known attack and compares them to the current flow of data. If a signature matches with the network activity, users and network administrators are notified. Anomaly based detection is where the system monitors current network traffic and compares it to a set baseline traffic. Again, if any unusual traffic occurs, members of the network are notified. In this research, new advancements in deep learning algorithms are used to bolster the defenses of digital networks. Neural networks are used to create a multi-class classifier, which will determine whether the network activity is a certain type of malicious attack or benign. We will use the CICIDS2017 dataset (Canadian Institute of Cybersecurity), which is a state-of-the-art network intrusion dataset composed of computer network activity, including multiple types of attacks such as DDoS, SQL Injection, and Brute Force. This research proposes a more precise network intrusion detection system (NIDS) to accurately detect malicious network activity. Better NIDSs will also prevent cybercrime and create a safer internet for all users.

Introduction

In April 2021, Colonial Pipeline's computers were attacked by hacking group DarkSide and forced to pay $4.4 million as ransom. Chaos ensued as gas supply to the east coast was blocked for several days. Cybersecurity attacks, such as network intrusions, can leak sensitive data, damage infrastructure, cause financial loss, and ruin a company's reputation. Network intrusion means an outsider has access to information that only people inside the network should have. The current method of detecting intrusions is via an Intrusion Detection System (IDS), which proactively sends an alert when an intrusion attempt is made. Although IDSs have been an industry standard for a while, they still have a few key flaws. For example, IDSs detect intrusions by checking for statistically deviant user activity. Over time, attackers can gradually change their activity such that malicious attacks are within the bounds of normal user activity. By using machine learning, attacks can be detected faster and more accurately, because machine learning can be leveraged for pattern recognition from historical data. In this research, the Canadian Institute for Cybersecurity's Intrusion Detection Evaluation Dataset (CICIDS2017) data set is used to train for 15 various attacks ranging from benign to DDoS to SQL injection, etc.

In a 2003 study, Sabhnani and Serpen acknowledged the fact that most intrusion detection systems only use one algorithm to detect various types of attacks. As a result, some attacks are detected with very low detection rates. The paper tests out nine different machine learning algorithms, but many had very low detection rates and high false alarm rates. All nine algorithms were unsuccessful at detecting U2R and R2L attacks at up-to-par levels. Additionally, the KDD 1999 Cup dataset was used, which is outdated and only covers four types of attacks (DoS, U2R, R2L, and Probing attacks). This experiment, however, does not test neural networks.

In a 2015 study, Zamani and Movahedi address the issue that many IDSs have high false positive and false negative rates. Furthermore, current methods do not adapt to the rapidly changing nature of malicious attacks. As a part of the study, the scientists reviewed a paper where Mukkamala et al proposes an intrusion detection system using artificial neural networks and Support Vector Machines (SVM). They aim to discover patterns that represent human behavior when creating anomaly recognizing classifiers. Training was done using the Radial Basis Function (RBF) as an activation function and an accuracy of 99.5% was reached. However, the SVM can only be used for binary classification, which means it cannot be used to classify multiple classes. Mukkamala et al's study is similar in approach to this research since machine learning methods are proposed as an alternative Intrusion Detection System for both. However, Mukkamala et al use SVM and RBF, which differs from this proposed approach.

Materials and Methods

Materials

To code the program, a Jupyter Notebook is used. This allows for all of the code to be stored locally and for the code to be accessible offline. The programming language of choice is Python due to its scalability and ease of use. Specifically, the NumPy library is used for more efficient processing of data in array form. Additionally, TensorFlow is used for its ability to create multi-layered neural networks. Lastly, the Pandas library is used for data analysis. The dataset used for this experiment is called CICIDS2017. This dataset contains 78 different features describing all sorts of information from packet size to packet rate.

Procedural Steps

Dataset and Data Pre-processing

The dataset used in this experiment is the CICIDS2017 Canadian Institute of Cybersecurity (CICIDS2017). This dataset is state-of-the art and covers all major types of attacks. To conduct the experiment, an Attack-network and Victim-network were implemented in the testbed. The Victim-network is a secure infrastructure with a Firewall, Router, switches, and multiple OSs. The Attack-network is a separate infrastructure with a router, switch, set of PCs with public IP, and various OSs needed for the attacks. Figure 1 shows all of the various components in the Victim-network.

	Machine	OS	IPs
Victim-Network	Servers	Win Server 2016 (DC and DNS) Ubuntu 16 (Web Server) Ubuntu 12	192.168.10.3 192.168.10.50-205.174.165.68 192.168.10.51-205.174.165.66
	PCs	Ubuntu 14.4 (32, 64) Ubuntu 16.4 (32-64) Win 7Pro Win 8.1-64 Win Vista Win 10 (Pro 32-64) Mac	192.168.10.19-192168.10.17 192.168.10.16-192.168.10.12 192.168.10.9 192.168.10.5 192.168.10.8 192.168.10.14-192.168.10.15 192.168.10.25
	Firewall	Fortinet	
Attackers	PCs	Kali win 8.1 Win 8.1 Win 8.1	205.174.165.73 205.174.165.69 205.174.165.70 205.174.165.71

Figure 1. Various components of Victim-network

In order to generate realistic background traffic, the data set has used the B-profile system to generate normal user traffic. The B-profile system is responsible for the benign traffic recorded in the dataset. The background behavior of 25 users based on the HTTP, HTTPS, FTP, SSH, and email protocols are extracted.

All the attacks were captured over a five-day span starting from 09:00 on Monday July 3 and ran continuously until 17:00 on Friday Jul 7. Figure 2 shows when each attack was executed. All of the network activity is recorded and labeled by the CICFlowMeter, which tracks over eighty network traffic features and is moved to the dataset.

Days	Labels
Monday	Benign
Tuesday	BForce,SFTP and SSH
Wednes.	DoS and Hearbleed Attacks slowloris, Slowhttptest, Hulk and GoldenEye
Thurs.	Web and Infiltration Attacks Web BForce, XSS and Sql Inject. Infiltration Dropbox Download and Cool disk
Friday	DDoS LOIT, Botnet ARES, PortScans (sS,sT,sF,sX,sN,sP,sV,sU, sO,sA,sW,sR,sL and B)

Figure 2. Attack schedule during five-day span

The data set is separated into eight different csv files, each covering different types of attacks. The dataset has 78 different features including information about the packets. The attacks are also labeled into 15

different categories ('BENIGN', 'Bot', 'DDoS', 'DoS GoldenEye', 'DoS Hulk', 'DoS Slowhttptest', 'DoS slowloris', 'FTP-Patator', 'Heartbleed', 'Infiltration', 'PortScan', 'SSH-Patator', 'Web Attack Brute Force', 'Web Attack Sql Injection', 'Web Attack XSS')

Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets
54865	3	2	0	12	0
55054	109	1	1	6	6
55055	52	1	1	6	6
46236	34	1	1	6	6
54863	3	2	0	12	0
54871	1022	2	0	12	0
54925	4	2	0	12	0
54925	42	1	1	6	6
9282	4	2	0	12	0
55153	4	2	0	37	0
55143	3	2	0	37	0
55144	1	2	0	37	0
55145	4	2	0	37	0
55254	3	3	0	43	0
36206	54	1	1	0	0
53524	1	2	0	0	0
53524	154	1	1	0	0
53526	1	2	0	0	0
53526	118	1	1	0	0
53527	239	1	1	0	0

Figure 3. Sample of the dataset

Some major attacks in the dataset include DoS, PortScan, and Web Attack.

Denial of Service (DoS): The attacker tries to shut down a network by flooding it with unnecessary traffic. The network cannot handle all of the extra traffic and users are unable to access the network. DoS attacks can be done on many platforms such as GoldenEye, Hulk, SlowHTTPTest, and Slowloris. A major type of DoS attack is Distributed Denial of Service (DDoS). This is where an attacker uses multiple computers to flood a network with unnecessary traffic.

PortScan: Attackers attempt to find open ports in a network and determine if they are sending or receiving data. Hackers send messages to the ports and the response decides whether the port is being used or open. Port Scanning can give information about weak points in the network and the services running on the ports.

Web Attack: Attackers aim for weak spots in websites to gain unauthorized access to confidential information or modify the website. This attack can appear in many forms. The most common are brute force, SQL injection, and Cross Site Scripting (XSS).

- Brute Force: Using trial and error to guess all possible combinations to passwords, keys, and hidden web pages.
- SQL Injection:Injecting malicious SQL code into an input field to access data from the backend database. This attack can reveal information such as Personally Identifiable Information (PII) and other confidential company data.
- Cross Site Scripting (XSS): Injecting malicious script into a vulnerable website.

In order to clean the data, we first imported the math, numpy, h5py, tensorflow, pandas, and time Python libraries. Then, we loaded the files into the database and read each one. Next, we printed all the labels in the dataset and found the frequency of each attack. Each attack was given a corresponding number.

This experiment will use a neural network since it is efficient at handling large amounts of data and yielding excellent results. In order to have a well-trained neural network, the data must be split between training and testing. The training data is inputted into the algorithm and given the expected output. This way, the algorithm learns about the data and adjusts itself accordingly. The testing data is used to determine the accuracy of the algorithm. The output of the testing data is compared to the expected output to measure the accuracy. We

assume that optimal results will happen when 80% of the data is assigned to testing and 20% is assigned to training. This gives enough training samples to the program and enough samples to test the accuracy of the program.

Multi-class Neural Network Classifiers

Paola and Schowengerdt state that a neural network has processing nodes where the values of its inputs are processed and an output is given. At the beginning of the network, there is an input layer where the data is inputted into the neural network. This data is passed onto a new layer of nodes to be processed and so on. The end of a neural network, where the output is given, is called the output layer. The processing layers between the input and output layers are called hidden layers.

Inside the nodes, the input values are added together and passed to an activation function. The result from this activation function is the output of that node. Each connection between nodes has a weight, and inputs are multiplied by the weight.

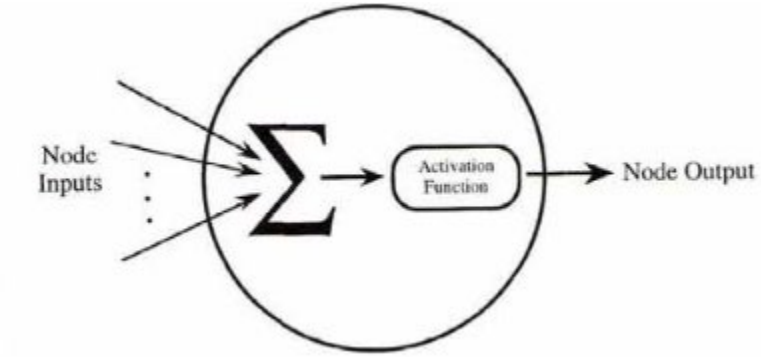


Figure 4. Inside a processing node

This experiment uses a multi-class classifier to determine what kind of attack the event was. The available data are called X variables and the outcome variable is called Y. The best prediction is called Y^. Each input is assigned a probability of being a certain class and then a classification rule determines what class the input is part of.

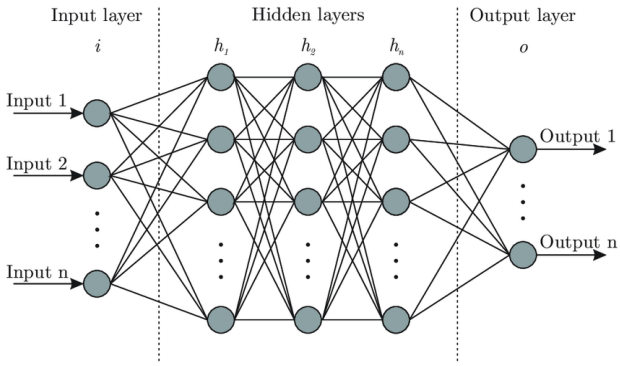


Figure 5. Design of a neural network

In the outputs of a neural network, the values can sometimes be negative and greater than one, which makes the probabilities hard to interpret. This is why the raw output values are sent to a softmax layer. We will use the softmax function, as seen in Formula 1, to give each output a scaled predicted probability between zero and one. The sum of all probabilities is equal to one and makes it easier to interpret the probability of each event.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

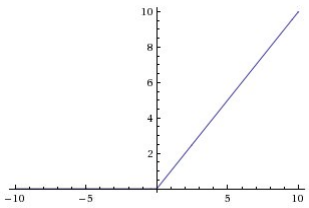
Formula 1: Softmax Function

To determine how well the neural network fits the data, we will use the cross entropy loss function. This function measures the accuracy of our outputs compared to the expected outputs. The sum of all of the cross entropy functions on all data points is called the Total Cross Entropy (shown in Formula 2) and shows the total error of the neural network. Since a negative logarithm is used in the loss function, correct outputs that are predicted with a low probability have extremely high cost. Likewise, correct outputs calculated with a high probability have a loss close to zero. By minimizing the Total Cross Entropy, the total error is minimized and the neural network is more accurate.

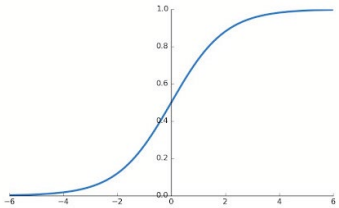
$$-\sum_{c=1}^{M} \text{Observed}_C \times \log(\text{Predicted}_C)$$

Formula 2: Total Cross Entropy Function

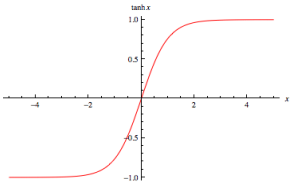
In our neural network, we will be using an activation function. An activation function is an equation that is attached to each neuron and gives an output. It is used when there are many inputs to a neuron that must be turned into an output. In this experiment, we will use a ReLU (Rectified Linear Unit), which is a piecewise-defined function where y=0 if x<0 and y=x if x>=0. In other words, the output is zero if the input is less than zero and the output is equal to the input if the input is greater than zero. Although there are many types of activation functions such as sigmoid and hyperbolic tangent function, they have flaws that the ReLU function covers. For example, the sigmoid and hyperbolic tangent function give values very close to zero for large negative inputs and one for large positive inputs, which can cause the gradient to vanish during backpropagation. The ReLU function allows for easier training and better performance.



Function 1: ReLU



Function 2: Sigmoid



Function 3: Hyper-

Neural Network Architecture

We will be testing various types of neural network models (changing number of layers, number of nodes, weightage of each node, etc.) and pick the model with the highest accuracy. The Keras Tuner package is used to test out various neural network architectures. Table 1 shows some Neural Network structures that will be tested. After running Keras tuner, the top ten architectures are trained with more epochs to increase the accuracy even further.

Table 1. Various Neural Network Structures

	# nodes						
Model Name	**(L = 1)**	**(L = 2)**	**(L = 3)**	**(L = 4)**	**(L = 5)**	**(L = 6)**	**(L = 7)**
3 Layers	30	20	-	-	-	-	-
4 Layers	40	30	20	-	-	-	-
5 Layers, Dropout #1	30	24	16	10	-	-	-
5 Layers, Dropout #2	50	40	30	20	-	-	-
5 Layers	50	40	30	20	-	-	-
6 Layers #1	50	40	30	25	20	-	-
6 Layers, Dropout #1	50	40	30	25	20	-	-
6 Layers #2	60	50	40	30	20	-	-
6 Layers, Dropout #2	60	50	40	30	20	-	-
7 Layers #1	50	40	30	25	20	15	-
7 Layers, Dropout	50	40	30	25	20	15	-
7 Layers #2	70	60	50	40	30	20	-

Results

We tested out hundreds of neural networks and compared our results to current non-machine learning NIDS and machine learning methods Ghorbani proposed in his paper. Our most accurate Neural Network classified the network attacks with an accuracy of 0.9933. Some classes in the dataset had very few training examples, which made classifying those classes very inaccurate.

The most accurate neural network had an architecture of 50, 150, 140, 145, 25, 40, and 15 nodes in each layer, respectively. The accuracy of this neural network was 0.9941. Figure 6 is the confusion matrix for this neural network

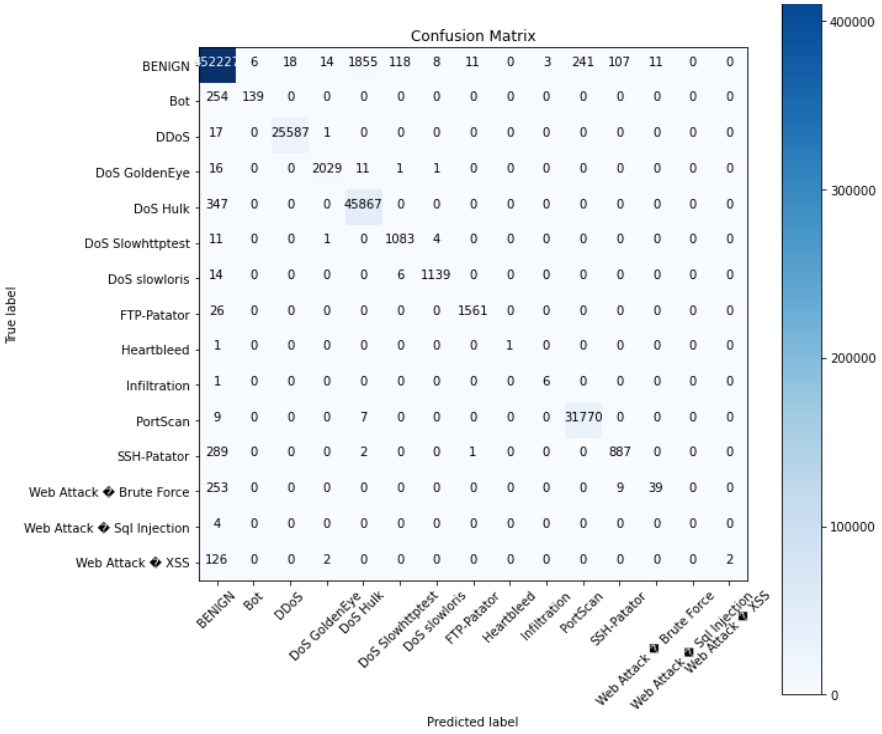


Figure 6. Confusion Matrix of the Most Accurate Neural Network

Another neural network had an architecture of 75, 90, 80, 90, 110, 120, 60, 45, 25, 150, and 15 nodes in each layer respectively. The accuracy of this neural network was 0.9932. Figure 7 is the confusion matrix for this neural network.

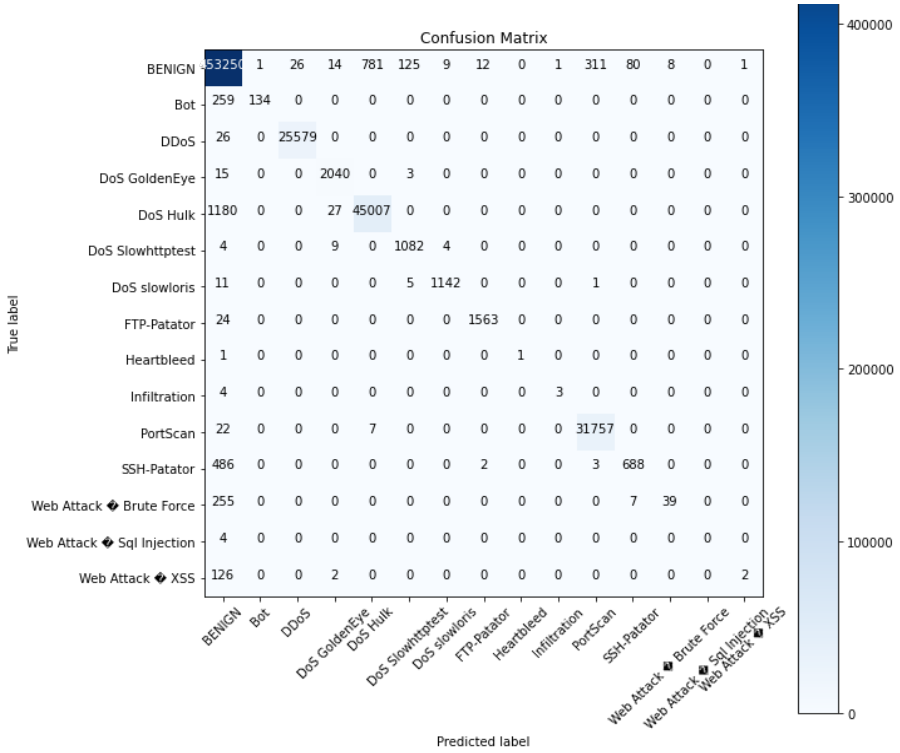


Figure 7. Confusion Matrix of the Second Most Accurate Neural Network

Heartbleed, Infiltration, Web Attack XSS, and Web Attack SQL Injection have very few training samples, as seen in Table 2, making it hard to accurately classify them while testing the neural network. The dataset is at fault here due it's lack of high fidelity data.

Table 2. Number of training examples in each class

```
label_frequency = df.groupby('Label').size()
print(label_frequency)
```

```
Label
BENIGN                          2273097
Bot                                1966
DDoS                             128027
DoS GoldenEye                     10293
DoS Hulk                         231073
DoS Slowhttptest                   5499
DoS slowloris                      5796
FTP-Patator                        7938
Heartbleed                           11
Infiltration                         36
PortScan                         158930
SSH-Patator                        5897
Web Attack  Brute Force            1507
Web Attack  Sql Injection            21
Web Attack  XSS                     652
dtype: int64
```

The neural network with architecture 50, 125, 75, 135, 150, 100, 15, 5, 70, 25, 50 was the most accurate at classifying the classes with few training samples. Heartbleed: ½, Infiltration: 6/7, Web Attack SQL Injection: 0/4, Web Attack XSS: 2/130. Figure 8 is the model's confusion matrix.

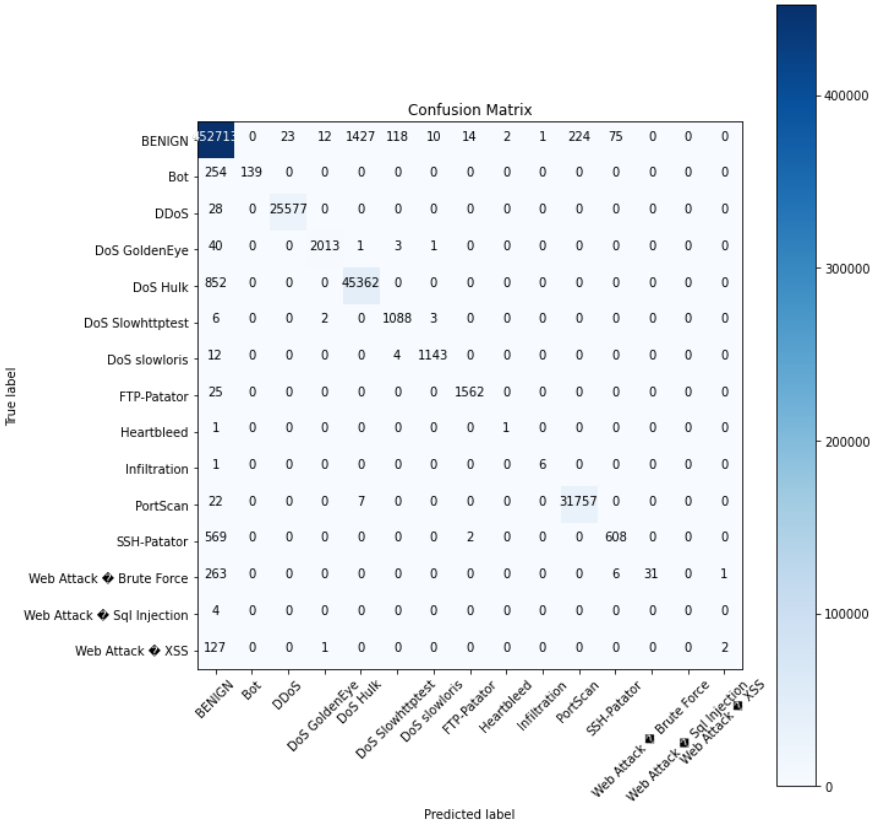


Figure 8. Confusion Matrix of Above Neural Network

Using Ghorbani's method, the most accurate algorithm was Quadratic Discriminant Analysis (QDA) with a precision of 0.97. However, the QDA algorithm ran in 18.79 seconds, while our neural network took 24.44 seconds. Table 3 shows all of the classification algorithms tested by Ghorbani.

Table 3. Accuracy of Ghorbani's Methods

Algorithm	Pr	Rc	F1	Execution (Sec.)
KNN	0.96	0.96	0.96	1908.23
RF	0.98	0.97	0.97	74.39
ID3	0.98	0.98	0.98	235.02
Adaboost	0.77	0.84	0.77	1126.24
MLP	0.77	0.83	0.76	575.73
Naive-Bayes	0.88	0.04	0.04	14.77
QDA	0.97	0.88	0.92	18.79

Discussion and Conclusion

Using deep learning to replace current network intrusion methods yield much more accurate results than current methods. This research can be expanded into other types of attacks and replace industry standards.

The most accurate neural network we found has an accuracy of 0.9941 and a precision of 0.9928, which is greater than the methods proposed by Ghorbani. By utilizing this new found research, advances can be made to the cybersecurity and network defense industry. The slow and unreliable methods of today will be

replaced by the new and more accurate methods of deep learning. In the future, we would like to find a dataset that covers more types of attacks and has more training samples for obscure types of network attacks.

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

Restrepo, Ronny. "Derivative of the Sigmoid Function - a Worked Example." *RSS*, 10 Aug. 2017, ronny.rest/blog/post_2017_08_10_sigmoid/.
"Hyperbolic Tangent." *From Wolfram MathWorld*, 23 Nov. 2021, mathworld.wolfram.com/HyperbolicTangent.html.
Bagli, E., Grandini, M., & Visani, G. (2020, August 13). *Metrics For Multi-Class Classification: An Overview.* https://arxiv.org/pdf/2008.05756.pdf
Dansbecker, "Rectified Linear Units (ReLU) in Deep Learning." *Kaggle*, 7 May 2018, www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning.
Ghorbani, A. A., Lashkari, A. H., & Sharafaldin, I. (2018). In Proceedings of the 4th International Conference on Information Systems Security and Privacy, Pages 108-116. *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization.* https://www.scitepress.org/Papers/2018/66398/66398.pdf

Mohavedi, M., & Zamani, M. (2015, May 9). *Machine Learning Techniques for Intrusion Detection.* https://arxiv.org/pdf/1312.2177.pdf

Paola, J. D., & Schowengerdt, R. A. (May 1997). *The Effect of Neural-Network Structure on a Multispectral Land-Use/Land-Cover Classification.* http://www.asprs.org/wp-content/uploads/pers/1997journal/may/1997_may_535-544.pdf

Restrepo, Ronny. "Derivative of the Sigmoid Function - a Worked Example." *RSS*, 10 Aug. 2017, ronny.rest/blog/post_2017_08_10_sigmoid/.
Sabhnani, M., & Serpen, G. (January 2003). *Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context.* https://neuro.bstu.by/ai/To-dom/My_research/Papers-0/For-research/D-mining/Anomaly-D/KDD-cup-99/CD4/mlmta03.pdf

Shukla, Lavanya. "Designing Your Neural Networks." *Medium*, Towards Data Science, 23 Sept. 2019, towardsdatascience.com/designing-your-neural-networks-a5e4617027ed.
Wood, Thomas. "Softmax Function." *DeepAI*, 17 May 2019, deepai.org/machine-learning-glossary-and-terms/softmax-layer.