Network intrusion detection system using deep neural networks

Avishek Saha, Noman Mohammad, Vimaljeet Singh

Abstract

In recent years, technological advancements and widespread internet services have led to an increase in piracy and cyber attacks on modern systems. As a result, it has become important to develop information security technologies that can detect new attacks. Intrusion Detection Systems (IDS) that use machine learning and deep learning techniques to detect network anomalies are one such technology. This paper proposes an advanced IDS with high network performance that uses a deep neural network algorithm to detect unknown attack packages. The attack detection is done by both binary classification methods, and the proposed system has achieved high accuracy rates (82.23%).

Introduction

Network Intrusion Detection Systems (NIDS) play a critical role in ensuring the security of modern computer networks. With the rise of cyber threats, NIDS has become an essential part of any organization's security infrastructure. However, traditional rule-based NIDS is becoming ineffective in detecting the latest types of network attacks. Therefore, Machine Learning (ML) algorithms are gaining popularity for developing NIDS that can detect and prevent network intrusions effectively. [1] In this project, we aim to design an ML-based NIDS to detect network intrusions in real-time. We also made an interface for the project using Flask so that we can do test runs and demo. The project will focus on exploring different neural network models to identify the most suitable model to detect network intrusion detection. The dataset used for the project will be the NSL-KDD dataset, which is a benchmark dataset for network intrusion detection research. ML-based models have the ability to learn from the training dataset and identify any unusual (anomaly) connection requests. The anomaly detection systems will usually detect and alert the security team about any suspicious network activity (or connection requests) or log it into Security Information and Event Management, enabling them to take proactive measures to prevent any possible security breaches later or track them. In the project we have done data pre-processing, exploratory data analysis, feature engineering, and model training which we will discuss as we move ahead.

Background

Network Intrusion

Network intrusion is an illegal activity that compromises network security and data security. It can occur due to hacktivism, stealing money or data, or spying. Network intrusion attacks can originate from individuals, major corporations, or even governments. The risks of network intrusion include corruption of data, financial loss, theft of data, operational disruption, and loss of reputation. Intrusion Detection System (IDS) and Intrusion Prevention System are two systems that can help in preventing network attacks. The Morris Worm is a well-known example of a network intrusion attack. [2]

Network intrusion detection refers to the process of monitoring network traffic for signs of unauthorized access or malicious activity. The goal is to identify suspicious behavior that could indicate an ongoing attack, security breach, or potential vulnerability. To detect network

intrusions, various techniques can be used, such as signature-based detection, anomaly detection, and behavior-based detection. Signature-based detection involves comparing network traffic to a database of known attack patterns, while anomaly detection uses statistical analysis to identify deviations from normal network behavior. Behavior-based detection involves monitoring the activity of users and systems on the network and identifying patterns of activity that suggest malicious intent. Once an intrusion is detected, there are several steps that can be taken to stop it. One option is to terminate the network connection associated with the attack. Another option is to block traffic from the attacker's IP address or range of addresses. It may also be necessary to disable compromised user accounts or take other remediation measures to prevent further damage. [3]
To prevent network intrusions from happening in the first place, it's important to have a comprehensive network security strategy that includes measures like firewalls, intrusion prevention systems, and user education. Regular security audits and vulnerability assessments can also help identify potential weaknesses that need to be addressed. Additionally, keeping software and operating systems up to date with the latest security patches can help prevent known vulnerabilities from being exploited.

Machine Learning & Network Intrusion Detection Systems

Machine learning can be a powerful tool for network intrusion detection because it enables the system to learn and adapt to new types of attacks over time. By analyzing large amounts of network data, machine learning algorithms can identify patterns and anomalies that may be indicative of a network intrusion. There are several ways that machine learning can be applied to network intrusion detection. One approach is to use supervised learning to train the system on known examples of malicious and benign network traffic. The system can then use this knowledge to classify new traffic as either malicious or genuine. Another approach is to use unsupervised learning to identify anomalies in network traffic that may be indicative of an intrusion. This can be especially useful for detecting previously unknown or zero-day attacks, which may not have a known signature or pattern. [4]

Reinforcement learning can also be used to improve network security over time. By providing the system with feedback on its performance, it can learn to adjust its behavior to better detect and prevent network intrusions. Overall, machine learning can provide a powerful tool for network intrusion detection, enabling systems to detect and respond to new types of attacks more quickly and effectively. However, it's important to note that machine learning is not a silver bullet and should be used in conjunction with other security measures, such as firewalls, intrusion prevention systems, and regular security audits. [5]

Methodology

Data Representation

The NSL-KDD dataset is a widely used benchmark dataset for research in network intrusion detection. It is an improved version of the KDD Cup 1999 dataset, which was created to evaluate intrusion detection systems for detecting network attacks. However, the KDD Cup 1999 dataset had some limitations, including an imbalanced class distribution, irrelevant features, and redundant records. To overcome these limitations, the NSL-KDD dataset was created.

The dataset contains 125,973 records, of which 22.5% are attack records. One of the most important benefits of the NSL-KDD dataset is that the number of records in the train and test sets is reasonable, making it affordable to run experiments on the complete set without randomly selecting a small portion.

This dataset has 41 features that can be classified into three main Categories: TCP connections features, Content features, and Traffic features.

ID	feature name	description	type
		TCP connections features	
1	duration	length (number of seconds) of the connection	continuous
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
3	service	network service on the destination, e.g., http, telnet, etc.	discrete
4	src_bytes	number of data bytes from source to destination	continuous
5	dst_bytes	number of data bytes from destination to source	continuous
6	flag	normal or error status of the connection	discrete
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong_fragment	number of ``wrong'' fragments	continuous
9	urgent	number of urgent packets	continuous
		Content features	
10	hot	number of ``hot'' indicators	continuous
11	num_failed_logins	number of failed login attempts	continuous
12	logged in	1 if successfully logged in; 0 otherwise	discrete
13	num_compromised	number of ``compromised'' conditions	continuous
14	root shell	1 if root shell is obtained; 0 otherwise	discrete
15	su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
16	num_root	number of ``root'' accesses	continuous
17	num_file_creations	number of file creation operations	continuous
18	num_shells	number of shell prompts	continuous
19	num_access_files	number of operations on access control files	continuous
20	num_outbound_cmds	number of outbound commands in an ftp session	continuous
21	is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
22	is_guest_login	1 if the login is a ``guest'' login; 0 otherwise	discrete
		Traffic features	
23	count	number of connections to the same host as the current connection in the past two seconds	continuous
24	dst_host_count	count of the connections having same dst host	continuous
25	serror_rate	% of connections that have ``SYN'' errors	continuous
26	rerror_rate	% of connections that have ``REJ'' errors	continuous
27	same_srv_rate	% of connections to the same service	continuous
28	diff_srv_rate	% of connections to different services	continuous
29	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
30	srv_serror_rate	% of connections that have ``SYN'' errors	continuous
31	srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
32	srv_diff_host_rate	% of connections to different hosts	continuous
33	dst_host_srv_count	count of connections have same dst host and using same service	continuous
34	dst_host_same_srv_rate	% of connections have same dst port and using same service	continuous
35	dst_host_diff_srv_rate	% of different services and current host	continuous
36	dst_host_same_src_port_rate	% of connection to current host having same src port	continuous
37	dst_host_srv_diff_host_rate	% of connections to same service coming from diff. hosts	continuous
38	dst_host_serror_rate	%of connection to current host that have an S0 error	continuous
39	dst_host_srv_serror_rate	%of connection to current host and specified service that have an S0 error	continuous
40	dst_host_rerror_rate	% of connection to current host that have an RST error	continuous
41	dst_host_srv_ rerror_rate	%of connection to the current host and specified service that have an RST error	continuous

Figure 1: Features of NSL KDD dataset

Exploratory Data Analysis

The dataset contains 22 types of attacks that can be grouped into 4 categories: Denial of Service (DoS), User-to-Root, Remote to Local (R2L), and Probing. DoS attacks overload the system to deny access to legitimate users. User-to-Root attacks aim to gain root access through

vulnerabilities. R2L attacks involve unauthorized access from a remote machine. Probing is an attempt to gather information about the network.

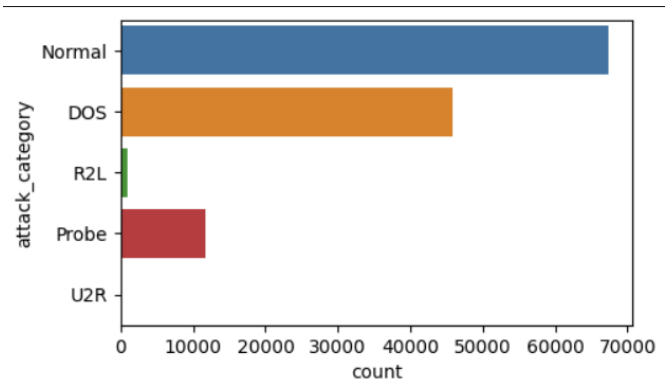

Figure 2: Classification of attack classes

Feature Engineering

The KDD CUP 99 dataset is free of missing values and noise, making it a clear dataset. However, it contains both numerical and text values, with the numerical values being large and potentially slowing down training and complicating processing. Additionally, deep neural network algorithms are not able to process text values directly, so the dataset must be preprocessed. The preprocessing steps in this model involve normalization and text mapping. To normalize the numerical attributes, the Z-score normalization method described in Equation 1 is used, which reduces the values and speeds up the training process while also reducing the space required.

$$Z = \frac{\chi - \mu}{\sigma} \qquad (1)$$

Where Z refers to the Z-score normalization, x refers to the values, μ refer to the mean of the sample and σ referred to the standard deviation of the sample. In another way, Converting the text attributes to numerical values have been executed using One Hot encoder easy handling with math equations as Figure 3 represent the execution of protocol type attribute (column in KDDCUP 99 dataset) on one hot encoder

Protocol type
tcp
tcp
udp
icmp
tcp
icmp

Protocol type tcp	Protocol type udp	Protocol type icmp
1	0	0
1	0	0
0	1	0
0	0	1
1	0	0
0	0	1

Figure 3: Explain One Hot Encoder on Protocol Type column

The number of features has been increased to 143 features rather than 41 features after executing one hot encoder on the categorical features..Using KBest technique find the useful predictors to reduce the dimension.

Model Training

Initially, a logistic regression model was utilized as a base model. Then, a neural network model was developed that included all the available features. Following that, a neural network model was constructed that incorporated only the most relevant features. Finally, an LSTM model was utilized for analysis.

Experiment & Results

Model Structures:

Neural Network with all features

Layer	Nodes	Activation Function
Input Layer	142	
Hidden Layer 1	50	ReLU
Hidden Layer 2	50	ReLU
Hidden Layer 3	30	ReLU

Multi Layer Perceptron with all features/selected features

Layer	Nodes	Activation Function
Input Layer	142/13	
Hidden Layer 1	30	tanh
Hidden Layer 2	40	tanh
Hidden Layer 3	50	tanh

LSTM

Layer	Nodes	Activation Function
Input Layer	142	
Hidden Layer 1	64	tanh
Hidden Layer 2	1	sigmoid

Experiment has done in these model by tuning hyperparameter such as
Number of layers: the number of hidden layers in the network.
Number of neurons: the number of nodes in each layer.
Activation function: the function used to introduce nonlinearity in the model.
Learning rate: the step size used in updating the weights of the model during training.
Dropout rate: the percentage of neurons that are randomly dropped out during training to prevent overfitting.
Batch size: the number of training examples used in each iteration of gradient descent.

Optimizer: the algorithm used to update the weights of the model during training, such as stochastic gradient descent or Adam.

Through a series of experiments, we have found that the MultiLayer Perceptron model with selected features provides the best test accuracy result compared to other models. This model achieved a test accuracy of 82.23%, which is the highest among all the models. Moreover, the precision score for this model was 0.823464, the recall score was 0.822658, and the F1 score was 0.819620.

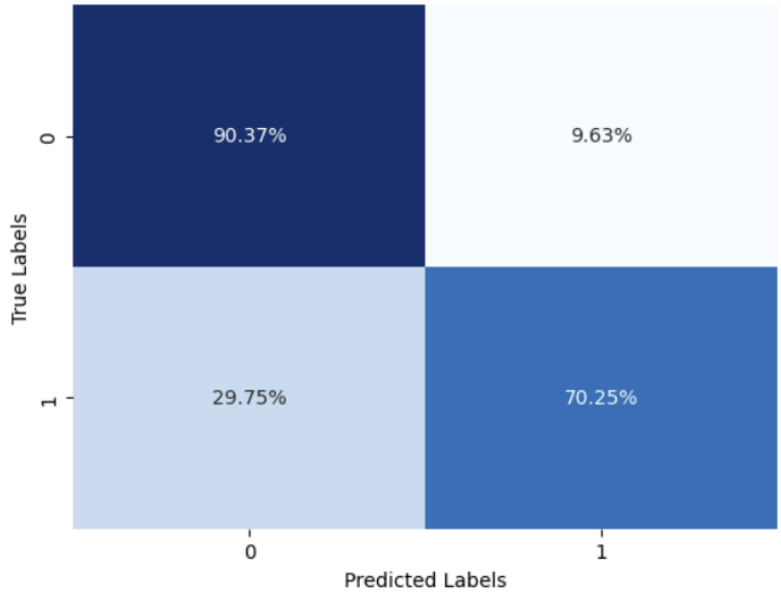

Figure 4: Confusion Matrix

Based on the onfusion matrix, it can be observed that the model predicted 12,165 true negatives and 6,381 true positives. However, it also misclassified 1,296 instances of the positive class as negative (false negatives) and 2,702 instances of the negative class as positive (false positives).
In summary, after testing and analysis, we concluded that the MultiLayer Perceptron model with selected features is the most effective in detecting and classifying attacks in our dataset. Overall, the model seems to be performing reasonably well, but there is room for improvement, especially in terms of reducing the false positive and false negative rates.

References

[1] Thomas M. Chen, in Computer and Information Security Handbook (Third Edition), 2013
[2] https://www.sunnyvalley.io/docs/network-security-tutorials/what-is-network-intrusion
[3] https://www.unb.ca/cic/datasets/nsl.html
[4]https://www.sunnyvalley.io/docs/network-security-tutorials/what-is-network-intrusion#:~:text=%E2%80%8B,access%20to%20the%20internal%20systems
[5]https://www.geeksforgeeks.org/intrusion-detection-system-using-machine-learning-algorithms/