Eye of Sauron: Using Neural Network for Anomaly Network Intrusion Detection

Arizona State University CSE 548

Derbique, Austin
aderbiqu@asu.edu

Leyba, Kirtus
kleyba@asu.edu

Schmidt, Ryan
raschmi4@asu.edu

March 18, 2021

Abstract

Machine learning based approaches to detecting anomalous or malicious network traffic are necessary in today's age with sophisticated attackers capable of evading typical firewall rules or signature-based matching of exploits. We examine using a feedforward neural network and self-organizing map to classify network whether traffic is benign or constitutes an attack. By evaluating these neural networks on a variety of training and test data, we can see how they perform and under which circumstances they perform best. This helps to provide a deeper understanding of the features that make for a successful machine learning based anomaly detection system when applied to real-world traffic.

Keywords— Neural Networks, Machine Learning, Anomaly Detection, Network Intrusion

I. Introduction

Today's networks are increasingly under attack. While stateful firewalls can control whether or not traffic gets blocked, they cannot determine whether any particular packet or connection is benign or malicious beyond examining the connection state combined with data from the current packet only [1]. As such, there is a need for utilities to examine these traffic flows to determine when a compromise might be happening. A typical approach to do this is to make use of an Intrusion Detection System (IDS). An IDS operates by examining the network traffic against a database of known exploit signatures, and raises alerts whenever traffic matches on of these signatures [2].

Signature databases fall short in detecting novel attacks, and modifications of attacks designed to evade the signatures. This is where anomaly detection comes into play. Because the malicious traffic often does not fit to the same patterns used for benign traffic, we can say it is anomalous. By using an anomaly detection system, we therefore are able to perform this classification of traffic that typical stateful firewalls and IDSes are unable to perform. Anomaly detection relies on Machine Learning (ML) to determine whether or not any particular network connection is malicious. In this paper, we will examine two ML approaches to anomaly detection and discover how well they perform.

II. System Models

A. System Model

We designed two neural networks, one Feedforward Neural Network (FNN) and one Self Organizing Map (SOM). An FNN consists of layers of neurons hooked together in a directed acyclic graph, so that data progresses from the input layer through the hidden layers, if any, and finally to the output layer. Due to the acyclic nature of the connections, information flows in only one direction. This is illustrated in Figure 1. To train an FNN, the technique we utilize is back propagation. Our training data is labelled with the correct responses, so by running the data through the FNN and comparing its output with the expected output, we can determine where the network needs adjustment. The back propagation algorithm takes these incorrect responses and calculates an error function, feeding the calculated value backwards through the network to adjust the parameters of the layers, so that next time it will generate more correct responses for that data. Over multiple training cycles, the network will converge on neuron weights that minimize the error. A major risk

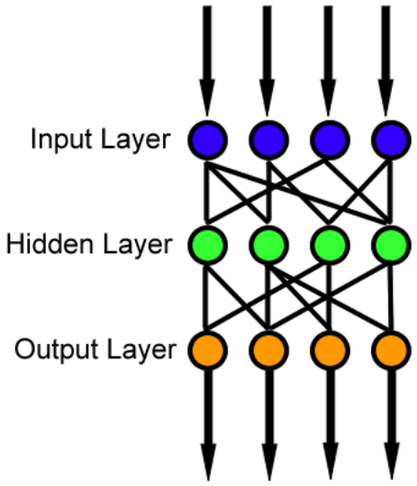

Figure 1: Diagram of a feedforward neural network. Data flows through the neurons in a single direction only. Image credit: Wikipedia [3].

of training FNNs is overfitting, in which the network learns the training data perfectly but loses any generalizations that allow it to accurately determine the categorization of unknown data.

Meanwhile, an SOM is typically used to reduce the dimensionality of a dataset. Commonly, it is used to create a 2-dimensional map of some higher-order data while retaining the general topology of the original data. During the training process, similar data is grouped together in a competitive process that does not require knowing the correct classifications of the data beforehand. This training mode is known as unsupervised training, because the neural network is creating its weights from training data that was not manually labelled with correct response beforehand. Instead, it extracts features based on the structure of the data. Once training is complete, new data is matched with the closest weight vector of the SOM to determine its classification.

B. Software

Various tools we have used are as follows:

- Python 3.8 is the language used to write our neural networks.
- The *tensorflow* and *keras* libraries for Python are used to provide the underpinnings of the neural networks.
- Additional Python libraries include *sklearn*, *pandas*, *numpy*, and *matplotlib* for parsing, preprocessing, and viewing the data.

C. Security Model

For an Intrusion Detection System to provide the most benefit, it must analyze traffic as it is flowing over the network. Our security model assumes that our neural networks are able to view this traffic stream live and be able to classify traffic as benign or malicious. If our neural networks discover malicious traffic, it would be up to some other system to determine what to do with it, such as blocking the traffic, sending alerts, or doing something more sophisticated such as various moving target defense schemes. In any event, what happens with the traffic after our neural network detects it is not part of our security model.

For our security model to work, we must be able to analyze all relevant traffic; if traffic bypasses our neural network then our classifications may be incorrect for the traffic that we do see, or we may never detect malicious traffic due to never being able to see it.

III. Project Description

Our project involves creating two neural networks, one FNN and one SOM, training them both with a variety of training data, and seeing how they perform under various scenarios with test data. As of the time of the midterm report, we are well underway with the data collection and analysis of the FNN, and we have begun laying the groundwork for the SOM.

A. Project Overview

We plan to work on the first and third parts outlined by the project specification: Anomaly Detection Accuracy with Unknown Attacks and Unsupervised Training for Detecting Network Attacks. Each of these parts has multiple components and deliverables associated with them, which are further broken down below.

By the midterm, we are planning to be underway with both parts, working on them simultaneously with different group members each taking the lead on their respective tasks. While we do not expect to complete all of the data collection by the time the midterm report is due, we should have a very good idea of what work remains and be close to the completion of each neural network.

B. Task 1: Set up Part 1 Framework

This task involves setting up the framework for Anomaly Detection Accuracy with Unknown Attacks by creating the python file, downloading the training data, and building the framework to save and load trained models for comparison purposes. Finally, the framework structure will be put on GitLab.

C. Task 2: Collect Part 1 Data

With the framework complete, we will need to train the network on subsets of the training data, then run the testing data against the trained models to see how they react. We will do this for multiple subsets and determine for each its accuracy and how it compares against the other subsets.

This task depends on Task 1 to be completed first.

D. Task 3: Complete Part 1 Deliverables

With the part 1 data collected, we will need to write the results in the report, create relevant figures, and create the demo video.

This task depends on Tasks 1 and 2 to be completed first.

E. Task 4: Set up Part 3 Framework

This task involves setting up the framework for Unsupervised Training for Detecting Network Attacks by creating the python file and downloading the training data, then putting the framework structure on GitLab.

F. Task 5: Collect Part 3 Data

With the framework complete, we will need to train the network. This will be done using the Day 5 data, with 50% of the data used for training and the other 50% used for testing to determine the network's ability to detect port scans. With the test data, we will determine the true positive, false positive, true negative, and false negative rates of the network.

This task depends on task 4 to be completed first.

G. Task 6: Complete Part 3 Deliverables

With the data collected, we need to format it for inclusion in the discussion section, including creating relevant figures or tables. Additionally, research will need to be made into the pros and cons of using unsupervised training for network attack detection. Finally, we need to create the demo video.

With exception of the research component, this task depends on tasks 4 and 5 to be completed first.

H. Task 7: Complete Midterm Report

At the time of the midterm, we need to evaluate how far along we are and how that compares to our projected timeline, then provide relevant discussion on our current progress and things we have learned.

I. Task 8: Complete Final Report

For the final report, we need to complete all deliverables and include them in the report. Additionally, we need to flesh out the shared sections, updating the introduction, system model, discussion, and conclusion with what we have learned.

This task depends on tasks 3 and 6 to be completed first.

J. *Task 9: Edit and Upload Demo Videos*

After recording both demo videos, we need to edit them to ensure they remain within the alloted time limit of 10 minutes each and to make the videos flow better. Once edited, they will each be individually uploaded to YouTube so that other class members can do their peer evaluations.

This task depends on tasks 3 and 6 to be completed first.

K. *Project Task Allocation*

Our team will work together on all tasks, however, we will allocate the following tasks to task leaders to lead the effort and find solutions to obstacles in the way of getting the task completed. The overall workload breakdown for each task is given in Table 1. We have elected **Austin Derbique** as the project lead.

Task	Task Leader
Set up Part 1 Framework	Austin Derbique
Collect Part 1 Data	Kirtus Leyba
Complete Part 1 Deliverables	Ryan Schmidt
Set up Part 3 Framework	Kirtus Leyba
Collect Part 3 Data	Ryan Schmidt
Complete Part 3 Deliverables	Austin Derbique
Complete Midterm Report	Kirtus Leyba
Complete Final Report	Austin Derbique
Edit and Upload Demo Videos	Ryan Schmidt

Table 1: The workload distribution of the project

L. *Project Timeline*

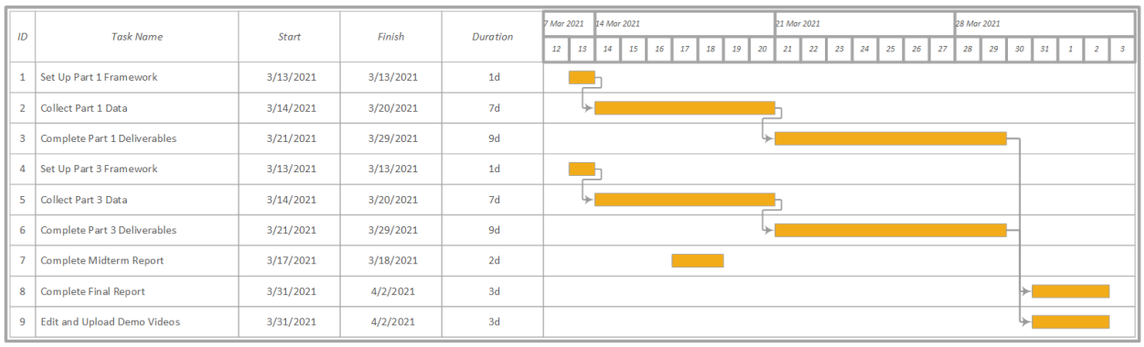

Figure 2: Gantt chart of project timeline

IV. Discussion of Anomaly Detection Accuracy with Unknown Attacks

We are ahead of schedule on part 1 of this project, having already completed the FNN and collected the data for it from a variety of training data subsets. Completing the deliverables for part 1 is underway, with many of the questions required in the report answered further in this discussion section. The demo video still needs to be created, edited, and uploaded.

Trained On	DoS Accuracy	Probe Accuracy	U2R Accuracy	R2L Accuracy
DoS	0.999	0.371	0.000	0.009
Probe	0.718	0.988	0.000	0.001
U2R	0.000	0.000	0.000	0.000
R2L	0.691	0.133	0.115	0.768
DoS, Probe	0.981	0.986	0.019	0.015
DoS, U2R	0.998	0.355	0.538	0.033
DoS, R2L	0.977	0.375	0.019	0.279
Probe, U2R	0.787	0.987	0.288	0.006
Probe, R2L	0.854	0.976	0.038	0.560
U2R, R2L	0.700	0.076	0.692	0.862

Table 2: This table shows the results of training the FNN model on different subsets of attacks (Shown in the Trained On column). The rest of the columns show how the model performed when tested on different attack categories.

How does the FNN model handle new attacks? We investigated the accuracy of the trained FNN model on new attacks by breaking the NSL KDD dataset into separate subsets for testing and training. An overview of these results is shown in table 2. The model always performs best on attacks it is trained to detect, which matches expectations. Interestingly, some new attack subsets of the data are detectible. For instance, we see that if we train a model to detect probe attacks, it is reasonable at also detecting DoS attacks (with an accuracy of 0.718).

What is the model's behavior in terms of accuracy for new attacks? We can further interpret these results by noting that some features in the data allow the model to infer new attacks in some cases even without training on those attacks of that specific type.

How does the change in attack types for training impact the model? When we trained on larger subsets, for instance (Probe, U2R), the model did better on the untrained attack categories than with a single training subset. This indicates that the model can learn some common attack patterns with more information rich data.

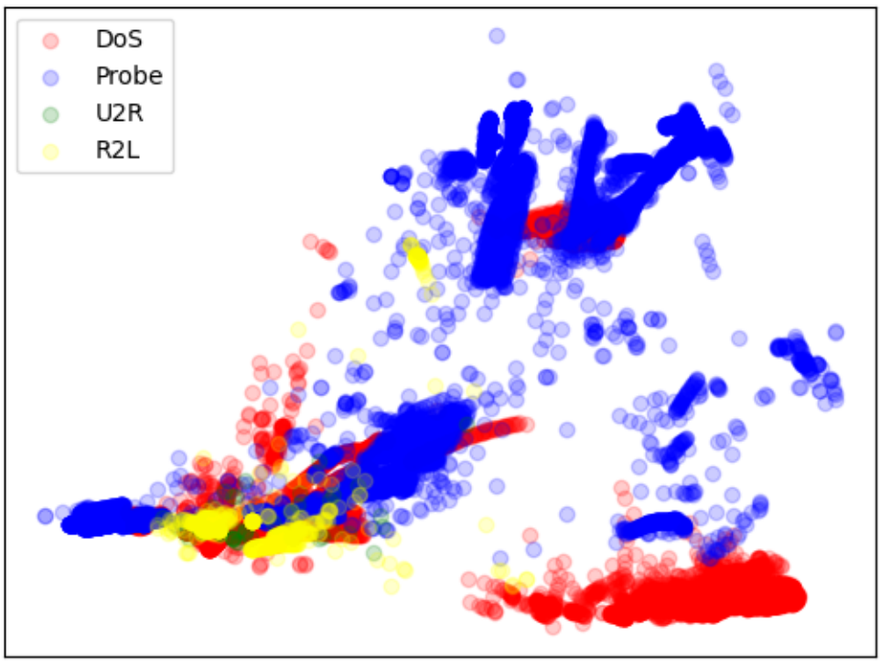

Figure 3: Principal Component Analysis (PCA) on the NSL KDD Train+ dataset. We use PCA to reduce the data dimensionality to visually compare the four main attack categories.

How do new attacks differ from trained attacks? To investigate what makes the attacks novel to the model we investigated the dataset. We performed principal component analysis (PCA) on the dataset as shown in figure 3 to visualize the differences between attack subsets. Notice that DoS and Probe seem to have a lot of overlap, aside from a special case of DoS attacks seen in the bottom right of the figure. Additionally, there is a large amount of overlap between U2R and R2L attacks. This reflects our other results in the sense that it appears that training the FNN model on similar attacks allows the model to detect novel attacks that are similar.

Does prediction accuracy relate to the attacks being similar? We found that if two attack subsets were similar then if we trained a model with one, we would be more effective at detecting the other. This is hugely important for the NSL KDD dataset because the U2R subset is much smaller than the other subsets. Note that the model trained purely on the U2R attack subset was by far the worst performing, but the model trained on both U2R and R2L attacks could actually detect U2R attacks with much higher accuracy. The richness in similar attack subsets can make a more robust intrusion-detection model!

V. Discussion of Unsupervised Training for Detecting Network Attacks

This part is not as far along as the FNN, as we have not yet completed developing and training the SOM. However, we are still on track to finish the SOM and collect data for it according to our original schedule shown in the timeline.

References

[1] Fortinet. *Stateful Firewall*. 2021. URL: `https://www.fortinet.com/resources/cyberglossary/stateful-firewall` (visited on 03/19/2021).

[2] Palo Alto Networks. *What is an Intrusion Detection System?* 2021. URL: `https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-detection-system-ids` (visited on 03/19/2021).

[3] Paskari. *Feed forward neural net.gif*. Nov. 2006. URL: `https://en.wikipedia.org/wiki/File:Feed_forward_neural_net.gif` (visited on 03/19/2021).