

2017

Using Self-Organizing Maps for Computer Network Intrusion Detection

Manuel R. Parrachavez

Follow this and additional works at: https://csuepress.columbusstate.edu/theses_dissertations



Part of the [Computer Sciences Commons](#)

Recommended Citation

Parrachavez, Manuel R., "Using Self-Organizing Maps for Computer Network Intrusion Detection" (2017). *Theses and Dissertations*. 293.

https://csuepress.columbusstate.edu/theses_dissertations/293

This Thesis is brought to you for free and open access by the Student Publications at CSU ePress. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of CSU ePress.

**USING SELF-ORGANIZING MAPS FOR COMPUTER
NETWORK INTRUSION DETECTION**

**Manuel R. Parrachavez
2017**

COLUMBUS STATE UNIVERSITY

USING SELF-ORGANIZING MAPS FOR COMPUTER NETWORK INTRUSION DETECTION

Copyright © 2017 Manuel R. Parrachavez

A THESIS SUBMITTED TO THE

HONORS COLLEGE IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS IN THE DEGREE OF

BACHELOR OF SCIENCE

TSYS SCHOOL OF COMPUTER SCIENCE

TURNER COLLEGE OF BUSINESS

BY

MANUEL R. PARRACHAVEZ

ABSTRACT

Anomaly detection in user access patterns using artificial neural networks is a novel way of combating the ever-present concern of computer network intrusion detection for many entities around the world. Anomaly detection is a technique in network security in which a profile is built around a user's normal daily activities. Profiles can be as following: successful access attempts; failed login attempts; access failures; and countless others. This data is collected and used as training data for a neural network.

There are many types of neural networks, such as multi-layer feed-forward network; recurrent networks; support vector machines; and others. For our study, we implemented our own self-organizing map (SOM), which we found to not be as heavily researched as other neural network approaches. Using the KDD Cup 99 dataset, we compared our own SOM implementation against other neural network implementations and determine the effectiveness of such an approach.

INDEX WORDS: Neural Networks, Self-Organizing Map, Anomaly Detection, KDD Cup 99

ABSTRACT

Anomaly detection in user access patterns using artificial neural networks is a novel way of combating the ever-present concern of computer network intrusion detection for many entities around the world. Anomaly detection is a technique in network security in which a profile is built around a user's normal daily actions. The data collected for these profiles can be as following: file access attempts; failed login attempts; file creations; file access failures; and countless others. This data is collected and used as training data for a neural network.

There are many types of neural networks, such as multi-layer feed-forward network; recurrent networks; support vector machines; and others. For our study, we implemented our own self-organizing map (SOM), which we found to not be as heavily researched as other neural network approaches. Using the KDD Cup 99 dataset, we compared our own SOM implementation against other neural network implementations and determine the effectiveness of such an approach.

INDEX WORDS: Neural Networks, Self-Organizing Map, Anomaly Detection, KDD Cup 99

TABLE OF CONTENTS

ABSTRACT	
LIST OF FIGURES	v
INTRODUCTION	1
PROJECT GOAL AND METHODOLOGY	1
ARTIFICIAL NEURAL NETWORKS IN ANOMALY DETECTION.....	2
SELF-ORGANIZING MAPS.....	4
EXPERIMENT AND RESULTS.....	6
CONCLUSIONS AND FURTHER WORK.....	12
BIBLIOGRAPHY.....	15

INTRODUCTION

TABLE OF FIGURES

Anomaly detection is a form of anomaly detection that is based off users' normal actions on a network system. Anomaly detection is done by observing the network and notifying the appropriate personnel when abnormal actions are being taken on the said network. Artificial neural networks (ANN) are able to excel in this type of environment because of the classification nature of anomaly detection. After being trained on testing data that was collected over a certain time, which is generally the most expensive and difficult part of a neural network based anomaly detection system, a trained neural network will be set on monitoring a network for misuse or irregularity in activity, thus classifying the traffic behavior as normal or abnormal. This allows for adaptability to new and different attacks if they do not fit the normal activity of the network.

FIGURE 1	SOM EXAMPLE	6
FIGURE 1	1 ST NETWORK	9
FIGURE 2	3 RD NETWORK	9
FIGURE 3	10 TH NETWORK.....	10

PROJECT GOAL AND METHODOLOGY

We created a Self-Organizing Map (SOM), a type of artificial neural network with unsupervised learning, using the Python programming language. We chose Python because of its wide array of deep learning libraries that were very helpful in implementing our SOM. Python also has a variety of tools available for data processing which needed to be done to our data before using it for training. We also chose the KDD Cup 99 dataset for our training and testing. It is widely used by researchers of neural network anomaly detection implementations which creates a benchmark for our SOM when compared to other implementations. The

INTRODUCTION

Anomaly detection is a form of Intrusion Detection that is based off users' normal actions on a network system. Anomaly detection is done by observing the network and notifying the appropriate personnel when abnormal actions are being taken on the said network. Artificial neural networks (ANN) are able to excel in this type of environment because of the classification nature of anomaly detection. After being trained on testing data that was collected over a certain time, which is generally the most expensive and difficult part of a neural network based anomaly detection system, a trained neural network will be set on monitoring a network for misuse or irregularity in activity, thus classifying the traffic behavior as normal or abnormal. This allows for adaptability to new and different attacks if they do not fit the normal activity of the network.

PROJECT GOAL AND METHODOLOGY

We created a Self-Organizing Map (SOM), a type of artificial neural network with unsupervised learning, using the Python programming language. We chose Python because of its wide array of deep learning libraries that were very helpful in implementing our SOM. Python also has a variety of tools available for data processing which needed to be done to our data before using it for training. We also chose the KDD Cup 99 dataset for our training and testing. It is widely used by researchers of neural network anomaly detection implementations which creates a benchmark for our SOM when compared to other implementations. The

dataset also comes with training and testing data which makes it very convenient for our use.

We will have to normalize the data before using it for training and testing. Due to our use of the

KDD Cup 99 dataset, we will have ample amount of other experiments to compare ours against.

Our goal was to have a reasonably accurate and novel solution to anomaly detection.

ARTIFICIAL NEURAL NETWORKS IN ANOMALY DETECTION

There are many different types of neural networks, and all of them have many situations where they are either more effective than, or not as effective, as another neural network. This section will provide an overview of the types of neural networks used by researchers for anomaly detection, and their advantages or shortcomings.

Multi-layer feed forward neural networks have a layered structure, allowing for a variety of complex tasks to be accomplished. They have the ability to find errors in the hidden layers by back-propagating the units of the output layer, therefore creating a more accurate model than a simple one-layer network (Krose & van der Smagt, 1996). One such use of this model used by the Georgia Technical Research Institute involved "4 fully connected layers, 9 input nodes and 2 output nodes (normal and attack)" (Philippe, 2001). Their model was successful in identifying each attack tested. MIT also used this model, which had k input nodes, $2k$ hidden nodes, and 2 outputs (normal and attack). Using the DARPA database, it detected 17 out of the 20 attacks, and one false alarm when monitoring UNIX host attacks through 30 keywords (Philippe, 2001). This model does not allow for feedback connections, thus it lacks the ability to dynamically alter and evolve to rapidly changing inputs. So, while this one might be the one of the simplest to create and use, its inability to be adapted to different events until it is retrained is a huge

drawback to this type of design, making this model very expensive to implement in a real-world situation (Cannady).

Recurrent neural networks have cycles built into them so that values – which can be either an extra set of input units that are fed some of the hidden unit activation values (an Elman network), or output values that are fed back into hidden units (Jordan network) – are run through the network as many possible times as until a stable point is reached (Krose & van der Smagt, 1996). The experiment results in 1998 and 1999 discovered the Elman network to be superior compared to Multilayer perceptron networks in anomaly detection. Recurrent networks were found to be quite expensive to train and required a large number of neural networks to be effective (Patcha & Park, 2007). A variation of this model is called the Long Term Short Memory Recurrent network. This network is trained on normal data, but is capable of performing live predictions for each time-step by “remembering” the preceding inputs and outputs, and adjusting its weights to the predicted output (Bontemps, Loi Cao, McDermott, & Le-Khac, 2017). In the experiment performed by researchers at University College Dublin, a 100% detection rate was achieved, but at the cost of a high false alarm trigger (Krose & van der Smagt, 1996).

Support Vector Machines are neural network models based more on a data mining approach. This allows them to detect new types of attacks because they do not depend on past patterns; however, the downside of this is the very high false positive rate (Patcha & Park, 2007).

Self-Organizing Maps (SOM) are networks that are trained to do their own mapping. A network like this is needed where there is no training data of inputs and the desired output pairs, for a given input the output is not known, and the only information is a set of input patterns (Kroese & van der Smagt, 1996). This method of training is called unsupervised learning. Self-Organizing Maps were used by Luc Girardin of UBILAB laboratory. The network performed clustering of network traffic and displayed the events on a 2D space for visualization, while those images were displayed to a network administrator who could see the deviations. This network was successful against IP spoofing; FTP password guessing; and network scanning and network hopping (Philippe, 2001). This technique was also used by Ohio University, through which they created a neural network for each of the network services being monitored. In their case, DNS and HTTP services were monitored (Patcha & Park, 2007).

We found that there is a lack of research on using SOMs, and we hypothesized that a SOM would be the novel solution to anomaly detection. With new attacks being invented at a rapid rate, a SOM will not have to be retrained with the new data, it will only have to know what normal behavior is to be able to detect anomalies.

SELF-ORGANIZING MAPS

Invented by Tuevo Kohonen, SOMs map multi-dimensional data into a two-dimensional grid or map, and is, in essence, a clustering algorithm (Kohonen & Honekka, 2007). In the application of a SOM to anomaly detection, normal behavior will be "clustered" together with abnormal behavior. This is done by adjusting the weight of the output neurons in order to become more like the data inputs. After a pre-determined amount of training iterations, with

each iteration causing the weights to be adjusted, clusters will be formed of similar data around neuron of best-fit through a method called competitive learning (Kohonen & Honekka, 2007).

The process of mapping to a two-dimensional space also allows for a simple visualization of highly complex data, making it much easier for a network administrator to see the patterns of all the data produced in a network. Otherwise, it would be incredibly difficult for an administrator to look at a vast set of unmapped data and deduce a meaningful conclusion. A SOM will be able to present this complex data in simpler and more visual way that the human brain can better comprehend (Guthikonda, 2005).

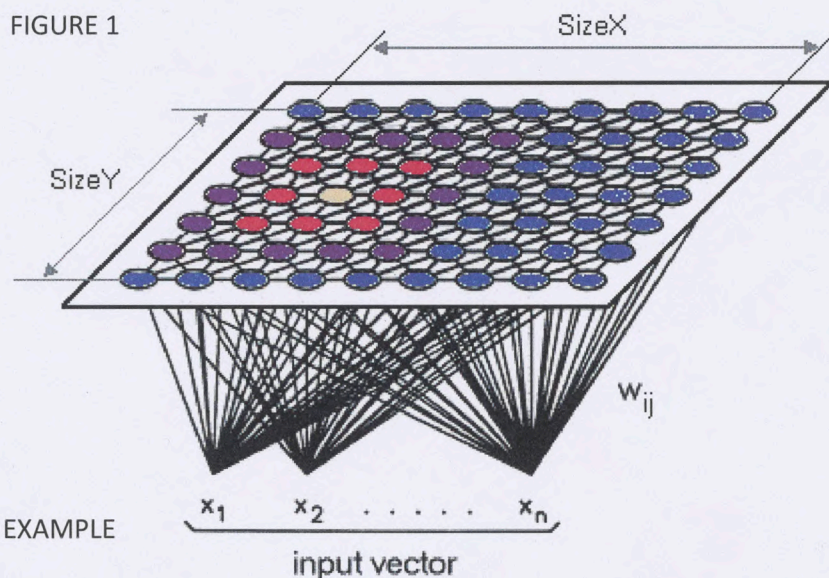
The SOM algorithm is presented as follow:

1. The SOM's nodes' weights are initialized. (A node is a location on the 2-D grid; there can be as many as desired, but they have to be fully connected to the input nodes.)
2. An input vector is randomly selected and given to the network.
3. Each network node is examined, and its weight is compared to the input vector. The winning node is called the Best Matching Unit (BMU). The BMU is usually calculated using the Euclidian distance formula.
4. Radius of the neighborhood is calculated, and is reduced every time-step using a decay function.
5. Any nodes within the radius of the BMU are adjusted to be more like the input vector. The closer the node is to the BMU, the more drastic the change is.
6. Repeat from step 2 for pre-determined iterations.

(Guthikonda, 2005)

As mentioned before, Self-Organizing Maps take advantage of a type of learning called unsupervised learning. Unsupervised learning means that a neural network is given only input data and no output data. The neural network does not need to know what the output of each input access pattern is, because it is instead finding the underlying inherent structure of the access patterns and clustering the into like groups.

While unsupervised learning is in itself a rare quality, what truly makes Self-Organizing maps unique is to take N-dimensional data and map it to a 2D map. The figure below is a visual representation of how the inputs dimension N are mapped to a 2D plane with each input vector being of N-dimensions. Each input vector is compared against other input vectors in a competitive learning style to get the resulting map (Joglekar, 2015).



EXPERIMENT AND RESULTS

For our experiments and implementation of our Self-Organizing Map, we used the following parameters:

- SOM Size: 7x7 map
- Training Iterations: 2,000
- Initial Learning Rate: .01
- Number of KDD99 Cup Training Records: 250,000
- Training Features: 38

We decided on a 7x7 map because of our hardware limitations, and the processing power that would be required in order to use a bigger map would be beyond our capabilities. We also believed that a smaller map, like a 3x3 or 5x5, would not be as effective with such a large dataset, while a 10x10 or 15x15 would be much too large for the computer equipment we have. Map size is also not an exact science in the realm of SOMs. There has been no clear method for choosing the map size other than it must not have more nodes than data points.

We also chose the training iterations to be 2,000, and an initial learning rate of .01, since these are default parameters in the training phase, and we did not want to put too many variables into the training process or training time. Our main focus of the experiments were SOM map size.

The first training run with the SOM, the entire kddcup.data_10_percent_corrected data set was used as the training set. With 494,021 records, the whole dataset, the amount of records caused the computer it was running on to crash and not work whatsoever. This was also only a 10 percent sampling of the original training set. With this in mind, different training amounts were tested in order to find a training amount that had a reasonable training time. Using 250,000 records were found to have an average training time of 9 minutes and 11

seconds, and was determined to be a fair time investment for each trained network for the purposes of the experiment.

At the beginning of the training, 38 of 42 features were selected creating a 38 dimensional training set. Normalization of the data was also required in order to get all of the data on a common scale. This was needed to stop certain data points from overtaking the other data points, and causing inaccurate clustering.

As a way to validate the clustering performance of the SOM, training of the SOM needs to occur multiple times with the same data. Comparisons of the clustering performance of the SOM was done visually. If the clustering of the data looks very similar across the produced SOMs, then the SOM could be considered validated.

The training was done 10 times with the 10th network chosen to take to testing. The SOMs show in FIGURES 1, 2, and 3 below are the networks that most closely resembled each other. There were several others that also looked very similar to these, but these ANNs were the closest matching. These SOMs are the visualization of two clusters representing "normal" and "attack" access pattern inputs in the SOM's output layer. The darker the shading, the more strongly the neuron has responded to "attack" access pattern units. The lighter the shading, the neuron more strongly responded to "normal" access pattern inputs.

FIGURE 2

Self_Organizing Map after 2000 iterations and a learning rate of 0.010000

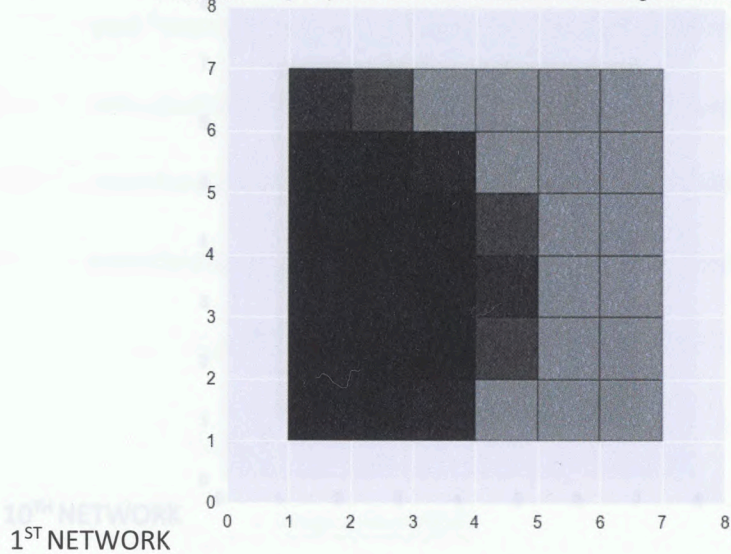
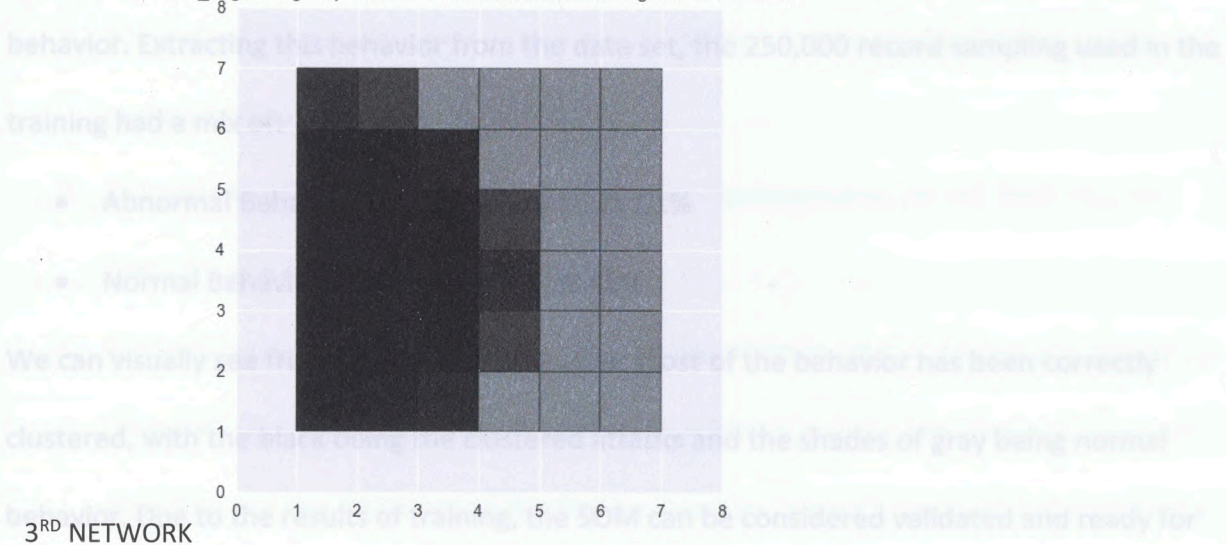


FIGURE 3

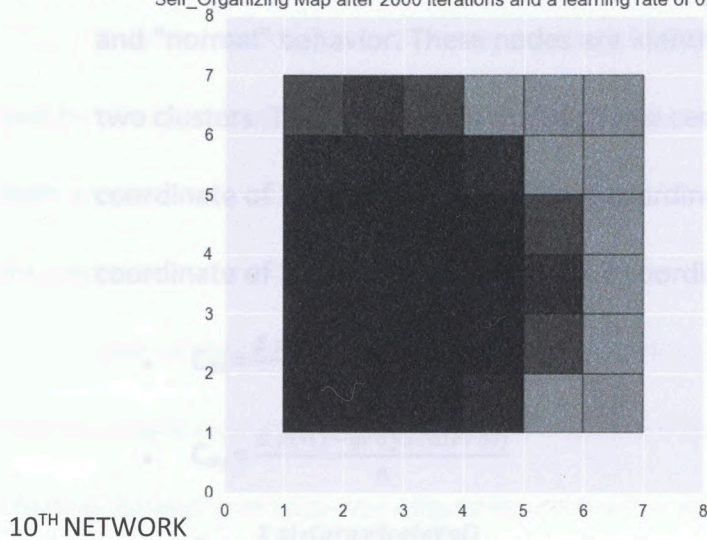
Self_Organizing Map after 2000 iterations and a learning rate of 0.010000



Using the records not employed in the training portion of the experiment, testing was done on the 10th produced SOM. For testing, the SOM was analyzed on the accuracy of the clustering it performed. Attack, Normal, False Positives, and False Negatives were kept tracked of. The outline of the testing process is presented below:

FIGURE 4

Self_Organizing Map after 2000 iterations and a learning rate of 0.010000



The use of the training data set allowed there to be a controlled mix of normal and abnormal behavior. Extracting this behavior from the data set, the 250,000 record sampling used in the training had a mix of:

- Abnormal Behavior: 178,775 records, 71.51%
- Normal Behavior: 71,225 records, 28.49%

We can visually see from the trained maps, that most of the behavior has been correctly clustered, with the black being the clustered attacks and the shades of gray being normal behavior. Due to the results of training, the SOM can be considered validated and ready for testing.

Using the records not employed in the training portion of the experiment, testing was done on the 10th produced SOM. For testing, the SOM was analyzed on the accuracy of the clustering it performed. Attack, Normal, False Positives, and False Negatives were kept tracked of. The outline of the testing process is presented below:

1. Find the two nodes in the 7x7 network that corresponds to the max output of "attack" and "normal" behavior. These nodes are identified by calculating the centroids of the two clusters. The formulas for finding these centroid nodes are given below. C_{ax} = X coordinate of "Attack" Centroid; C_{ay} = Y coordinate of "Attack" Centroid; C_{nx} = X coordinate of "Normal" centroid; C_{ny} = Y coordinate of "Normal" centroid.

- $$C_{ax} = \frac{\sum xi*(1-grayScaleVal)}{n}$$

- $$C_{ay} = \frac{\sum xi*(1-grayScaleVal)}{n}$$

- $$C_{nx} = \frac{\sum xi*(grayScaleVal)}{n}$$

- $$C_{ny} = \frac{\sum xi*(grayScaleVal)}{n}$$

2. For each of the records used in testing, find its Best Matching Unit (BMU) and its corresponding coordinates on the SOM.
3. Calculate the Euclidean distance between the corresponding record BMU and the "Attack" and "Normal" centroids.
4. If Attack is closest to "Attack" centroid, it is classified as attack; If Normal is closest to "Normal" centroid, it is classified as normal; if Attack is closest to "Normal" centroid, it is classified as False Negatives; if Normal is closest to "Attack" centroid, it is classified as False Positives.

Using these steps and the testing portion of the dataset, a 100% accurate clustering rate was achieved. There were no false negatives or false positives detected, and all attacks and normal behaviors were correctly clustered.

CONCLUSION AND FURTHER WORK

It can be seen that the Self-Organizing Map has a promising future in anomaly detection, and by extension computer network intrusion. However, there are some drawbacks of using such an approach, and many ways the method proposed in this paper could be improved on as discussed below.

One of the most obvious obstructions of using and SOM as an intrusion detection tool is that it is really expensive to train. For the research conducted in this paper, only half of the training dataset provided was able to be used without crashing the computer; this training dataset was also only 10% of the original training dataset. Using the whole training data set would have resulted in a training size of 4,940,220 records. This would have been far too much for our resources to handle, and for a large corporation, it might be an expensive investment. While it might be an expensive investment, if a corporation is looking for solution to a problem that can cause massive financial losses, the corporation would be willing to make that investment. In fact, computation time is considered the biggest drawback for SOMs in general (Palomo, Dominguez, Luque, & Munoz, 2008). This brings up the biggest barrier to using an SOM as part of an intrusion detection system, training data gathering. This is by far the most expensive process of the SOM training algorithm, because the organization that would try to implement this would have to collect their own data. A dataset found online would not be an accurate description of the network patterns that would occur on their own network, and therefore be completely useless to the organization. Data collecting is not an easy or cheap task to accomplish for a large organization and might be the biggest barrier to entry for this approach.

Personnel would also need to be trained in reading the produced Self-Organizing map. This is because the map does not produce results that are easy to read, or concrete numbers that can be easily understood. A network administrator will have to be tasked with looking the SOM and correctly inferring what the clusters mean.

In future work the following improvements and tasks would need to be done. Further techniques validating the produced SOM in order to have more confidence in the results. A more powerful machine would be required in order to process more data, and could make it possible to try increased complex configurations of the SOM like map size; training iterations, and training set size. More techniques that could analyze the clusters in order to create a higher confidence in the produced SOM. These other validation techniques could be K-Fold validation and the silhouette coefficient. These techniques would be very helpful if using unlabeled data, because these validations techniques were specifically created in order to validate unsupervised learning algorithms, which SOMs are a part of.

Despite all these criticisms, this implementation of Self-Organizing Maps performed remarkably well, even when compared to other neural network anomaly detection implementations. This SOM implementation would be tied for highest performing in clustering with accuracy when compared to these prior mentioned neural networks that include recurrent neural networks, multi-layer perceptron neural networks, support vector machines, and other SOMs. It is on par with the performance of a Bayesian network classifier by M. Bode that also used the KDD Cup 99 dataset (Hodo, Bellekens, Hamilton, Tachtatzis, & Atkinson). A 100% accurate clustering is a very good sign that this approach to anomaly detection in computer

network intrusion detection is a valid approach, and worth further pursuing with the proposed additions and improvements.

- McMill, J., & Le-Elbaz, N.-A. (2017). *Collective Anomaly Detection based on Long Short Term Memory Recurrent Neural Network*.
- Cannedy, J. (n.d.). *Artificial Neural Networks for Malware Detection*. Fort Lauderdale: Nova Southeastern University.
- Depron, O., Topalac, M., Anarim, E., & Giliz, M. K. (2006). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, 713-722.
- Guthikonda, S. M. (2005). *Kohonen Self-Organizing Maps*. Wittenberg University.
- Hodo, E., Ballekens, X., Hamilton, A., Tachtatzis, C., & Atkinson, R. (n.d.). *Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey*. University of Strathclyde and University of Abertay Dundee.
- Joglekar, S. (2015, November 28). *Self-Organizing Maps with Google's TensorFlow*. Retrieved from Sachin Joglekar's Blog: <https://codexsachin.wordpress.com/2015/11/28/self-organising-maps-with-googles-tensorflow/>
- Kohonen, T., & Honkela, T. (2007). Kohonen Network. *Scholarpedia*, 1568.
- Kroza, B., & van der Smagt, P. (1996). *An introduction to Neural Networks*. Amsterdam: The University of Amsterdam.
- Palomo, E. J., Dominguez, E., Lopez, R. M., & Munoz, J. (2008). A Competitive Neural Network for Intrusion Detection Systems. *Modelling, Computation and Optimization in Information Systems and Management Sciences*, 530-537.
- Patcha, A., & Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 3443-3470.
- Philippe, J. (2001). *Application of Neural Networks to intrusion*. SANS Institute.

Bibliography

- Bontemps, L., Loi Cao, V., McDermott, J., & Le-Khac, N.-A. (2017). Collective Anomaly Detection based on Long Short Term Memory Recurrent Neural Network.
- Cannady, J. (n.d.). *Artificial Neural Networks for Misuse Detection*. Fort Lauderdale: Nova Southeastern University.
- Depren, O., Topallar, M., Anarim, E., & Ciliz, M. K. (2006). An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, 713-722.
- Guthikonda, S. M. (2005). *Kohonen Self-Organizing Maps*. Wittenberg University.
- Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., & Atkinson, R. (n.d.). *Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey*. University of Strathclyde and University of Abertay Dundee.
- Joglekar, S. (2015, November 28). *Self-Organizing Maps with Google's TensorFlow*. Retrieved from Sachin Joglekar's Blog: <https://codesachin.wordpress.com/2015/11/28/self-organizing-maps-with-googles-tensorflow/>
- Kohonen, T., & Honekla, T. (2007). Kohonen Network. *Scholarpedia*, 1568.
- Krose, B., & van der Smagt, P. (1996). *An introduction to Neural Networks*. Amsterdam: The University of Amsterdam.
- Palomo, E. J., Dominguez, E., Luque, R. M., & Munoz, J. (2008). A Competitive Neural Network for Intrusion Detection Systems. *Modelling, Computation and Optimization in Information Systems and Management Sciences*, 530-537.
- Patcha, A., & Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 3448-3470.
- Philippe, J. (2001). Application of Neural Networks to Intrusion. *SANS Institute*.

Committee Member

John Barlow

Date

Nov 17, 2017

Dr. John Barlow

Committee Member

Jianhua Yang

Date

11/19/17

Dr. Jianhua Yang

Honors College Dean

Cindy Ticknor

Date

11/20/17

Dr. Cindy Ticknor

USING SELF ORGANIZING MAPS FOR COMPUTER NETWORK INTRUSION DETECTION

By

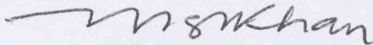
Manuel R. Parrachavez

A Thesis Submitted to the

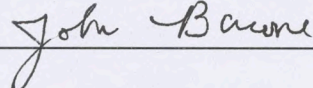
HONORS COLLEGE

In Partial Fulfillment of the Requirements
for Honors in the Degree of

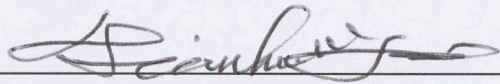
BACHELOR OF SCIENCE
COMPUTER SCIENCE
TURNER COLLEGE OF BUSINESS

Thesis Advisor  Date 11/14/17

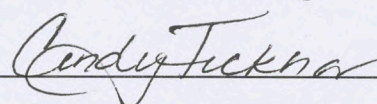
Dr. Shamim Khan

Committee Member  Date Nov. 14, 2017

Dr. John Barone

Committee Member  Date 11/14/2017

Dr. Jianhua Yang

Honors College Dean  Date 11/30/17

Dr. Cindy Ticknor

