5-12-2023

Pruning GHSOM to create an explainable intrusion detection system

Thomas Michael Kirby
Mississippi State University, kirbyt033@gmail.com

Pruning GHSOM to create an explainable intrusion detection system

By

Thomas Michael Kirby

Approved by:

Shahram Rahimi (Major Professor)
Sudip Mittal
Ioana Banicescu
Andy Perkins
T.J. Jankun-Kelly (Graduate Coordinator)
Jason M. Keith (Dean, Bagley College of Engineering)

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science
in the Department of Computer Science and Engineering

Mississippi State, Mississippi

May 2023

Name: Thomas Michael Kirby

Date of Degree: May 12, 2023

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Shahram Rahimi

Title of Study: Pruning GHSOM to create an explainable intrusion detection system

Pages of Study: 45

Candidate for Degree of Master of Science

Intrusion Detection Systems (IDS) that provide high detection rates but are black boxes lead to models that make predictions a security analyst cannot understand. Self-Organizing Maps (SOMs) have been used to predict intrusion to a network, while also explaining predictions through visualization and identifying significant features. However, they have not been able to compete with the detection rates of black box models. Growing Hierarchical Self-Organizing Maps (GHSOMs) have been used to obtain high detection rates on the NSL-KDD and CIC-IDS-2017 network traffic datasets, but they neglect creating explanations or visualizations, which results in another black box model.

This paper offers a high accuracy, Explainable Artificial Intelligence (XAI) based on GHSOMs. One obstacle to creating a white box hierarchical model is the model growing too large and complex to understand. Another contribution this paper makes is a pruning method used to cut down on the size of the GHSOM, which provides a model that can provide insights and explanation while maintaining a high detection rate.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The use of Artificial Intelligence (AI) in cyber-defense solutions, particularly Intrusion Detection Systems (IDS), has been gaining traction to protect against a wide range of cyber attacks. While these AI models have high detection rates, high false positive and false negative rates can dissuade a security analyst from using an AI enabled IDS [30]. Further, many of the AI and deep learning methods are black boxes, meaning a security analyst will have little to no explanations and clarifications on why a model made a particular prediction. With the rise in cyber attacks on critical infrastructure, government organizations, and business networks, there is a pressing need for an explainable, automated detection system that can provide real-time aid to an analyst.

Intrusion Detection Systems are generally utilized as part of a larger cybersecurity defense effort at an organization generally located in a Cyber-Security Operations Center (CSoC). These systems monitor networks and automate attack detection by comparing network activity to the signature of known attacks or by detecting behavior that is anomalous to benign network patterns [46]. Through these methods, a security analyst can use an IDS to detect improper use, unauthorized access, or the abuse of a network. Analysts can then create mitigating strategies to minimize damages and costs of the malicious behavior. The usefulness and cost effectiveness of IDS have therefore been the subject of much research [3, 58].

Previous work in AI enabled IDS has generally focused on improving detection rates while limiting false positives and false negatives. These techniques have been effective at achieving high detection rate, but have failed to provide explanations for their computed predictions. Without the ability to understand how a model reached a decision and which features were relevant to the decision computation, a security analyst will give less credence to these AI enabled IDS. Opaque Deep Learning methods in particular, can be considered as black boxes providing no explanations and feature relevance information, severely limiting adoption in real world cyber-defense scenarios [25].

A potential solution to this problem is to research and develop Explainable Intrusion Detection Systems (X-IDS) based on current capabilities in Explainable Artificial Intelligence (XAI) [1, 38]. The guidelines proposed by the Defense Advanced Research Projects Agency (DARPA) indicate that to be explainable, an AI should explain the reasoning for its decisions, characterize its strengths and weaknesses, and convey a sense of its future behavior [12]. An X-IDS that is transparent in its behavior and decision making process, will empower a security analyst to make better informed actions, understand the feature composition of a prediction, help CSoCs defend from known attacks, and quickly understand zero-day attacks. To address this need, an X-IDS using Growing Hierarchical Self Organizing Maps (GHSOMs) is implemented.

Data collected from modern networks contain potentially hundreds of different features about the traffic flow, operating systems, network protocols, and other metadata. SOMs work by representing this high dimensional data on a 2-dimensional plane. This also includes maintaining the topographical relationship of the data by grouping similar data [19]. GHSOMs go a step further and create a hierarchical tree structure of multiple SOMs, capturing more information from complex

input data [10]. Through SOM visualization techniques, a security analyst can view both global and local explanations about a potential attack rather than an opaque prediction generated by a black box model. However, the growing process of GHSOM results in a structure many layers deep and consisting of thousands of individual SOMs. To gain understanding from such a large structure would be daunting, so a pruning process of the GHSOM will be done to make the exploration space more navigable and understandable for the security analyst.

As the need for explainable cyber-defense systems increase and to address the lack of XAI research in the field of IDS, the main objective of this paper is to create an explainable IDS while maintaining high detection accuracy. To accomplish this the high detection accuracy GHSOM can achieve will be used along with pruning techniques to effectively visualize the tree structure and provide explanations for why a prediction was made. The goal in this paper is to increase trust in IDS and help CSoCs defend from attack through the use of explainable insights.

CHAPTER 2

RELATED WORK

2.1 Intrusion Detection Systems

An intrusion refers to an action that obtains unauthorized access to a network or system [9]. Intrusions can be characterized by a violation of Confidentiality, Integrity, or Availability (CIA). An IDS consists of tools, methods, and resources that help a CSoC protect an organization [2, 32].

IDS can be classified as either a host-based IDS or network-based IDS. Host-based IDS are placed on a host system and monitor host activity, incoming and outgoing network traffic [22]. Network-based IDS are built to survey and protect a network of hosts from intrusion [36]. In addition, IDS can also be categorized into operation-based classes, such as signature, anomaly, and hybrid. Signature-based IDS operate by preventing known attacks from accessing a network. The IDS compares incoming network traffic to a database of known attack signatures. Notably, this method has difficulty in preventing *zero-day* attacks [49]. Anomaly-based IDS look for patterns in incoming traffic to recognize potential threats and leverage complex AI models [6, 31]. A significant drawback of this approach is the the tendency for such systems to categorize legitimate, unseen behavior as anomalous. Hybrid-based IDS incorporate the design philosophy of both signature-based and anomaly-based IDS to improve the detection rate while minimizing false positives [40, 52].

Current AI enabled anomaly-based IDS can be further divided into black box and white box models. White box models are considered *easy to understand* by an expert. This allows the expert to analyze the decision process and understand how the model renders its decision. This ”semi-transparent” property allows white box models to be deployed in decision sensitive domains, where auditing the decision process is a requirement. White box models may use regression-based approaches [51], decision trees [28], and SOMs [21]. Black box models, on the other hand, have an opaque decision process. This opaqueness property makes establishing the relationship between inputs and the decision difficult, if not outright impossible. Black box models comprise nearly all the AI enabled state-of-the-art approaches for IDS, as the focus is traditionally on model performance, not explainability. Examples of black box models are Isolation Forest [26], One-Class SVM [48], and Neural Networks [61].

2.2 Explainable AI

As previously stated, state-of-the-art approaches for IDS, as well as machine learning as a whole, focus on model performance through the lens of model accuracy. This focus on model accuracy has driven the development further away from modeling approaches that are transparent or have methods of explainability. In turn, this creates a separation between model inference and *understanding* model inference, which gives the inability to confirm model fairness, privacy, reliability, causality, and ultimately trust.

The notion of XAI dates back to the 1970s. Moore et al. [35] surveyed works from the 1970s to the 1980s, detailing early methods of explanations. Some early explanations consisted of canned text and code translations, such as the 1974 explainer MYCIN [50]. DARPA provides a more

current definition of XAI by defining XAI as systems that are able to explain their reasoning to a human user, characterize their strengths and weaknesses, and convey a sense of their future behavior [12]. In turn, the system offers some form of justification for its action, leading to more trust and understanding of the system. The explanations from an XAI system help the user not only in using and maintaining the AI model, but also helping users complete tasks in parallel with the AI system. Tasks can include doctors making medical decisions [15, 24, 50], credit score decisions [7], detecting counterfeit banknotes [14] or CSoC operators defending a network [8, 12].

2.2.1 Explainable Intrusion Detection Systems

Explainable Intrusion Detection Systems (X-IDS) are still an emerging sub-genre in the field. The need for explainability in IDS is becoming increasingly necessary both to warrant further application in decision sensitive domains, as well as to supplement and empower existing knowledge techniques (e.g. data mining, rule-based development) that black boxes obfuscate. The users need to be confident in the predictions or recommendations computed by an IDS. Understandable explanations allow users to perform their tasks correctly. The stakeholders of an IDS (e.g. CSoC operators, developers, and investors) are individuals who will be dependent on the performance of the system. CSoC operators will be performing defensive actions based on prediction and explanation results. Developers can use explanations to fortify the model in areas where it is weak. Investors may need explanations to help them in making budgeting decisions for their company.

The current literature consists of many different black box and white box models being used alongside explanation techniques. Common explainer modules for black box models are Local Interpretable Model-agnostic Explanations (LIME) [47], SHapley Additive exPlanations (SHAP)

[27], and Layer-wise Relevance Propagation (LRP) [4]. Modern techniques for explaining black box models consist of creating surrogate models that generate explanations either locally or globally. Other methods involve propagating predictions backwards in a Neural Network or decomposing a gradient. More novel approaches have also experimented with making datasets explainable [17] or making GUIs for explainable systems [57].

2.3 Self-Organizing Map Algorithms

This section will outline the Self-Organizing Map algorithm along with improved variant algorithms. A brief overview of the steps of each algorithm, its previous use in IDS literature, and the disadvantages of each approach will be outlined.

2.3.1 Self-Organizing Maps

SOMs are an unsupervised learning technique that has been a commonly used method to make IDS due to there effectiveness at detecting anomalies and visualizing input data. [23,33,43,59,60]. They were first introduced in [19,20], mapping high-dimensional input data to a low-dimensional and topologically accurate map. This is done by first initializing a two dimensional set of neurons with weights equal to the dimensions of the input data and edges connecting the neurons. The training data is then iterated through, and for each data point, the neuron with the weights closest to the data point becomes the best matching unit (bmu) or winning neuron. The bmu then updates its weights and the weights of neighboring neurons to be closer to the corresponding data point. This results in a map where high dimensional data points close to each other having winning neurons also close to each other in lower dimensional space. Figure 2.1 shows the training process of SOMs [55]. Through this method researchers creating IDS with SOMs create normal patterns of

network traffic and detect any anomalous events. Disadvantages arise when using SOMs, namely having to predetermine the correct network size and the inability to accurately model complex datasets.

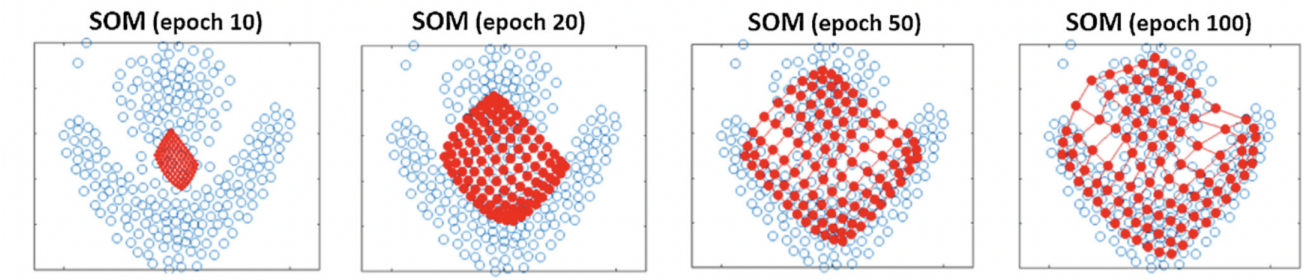


Figure 2.1: This shows the training process of SOM. The blue circles represent the input data vectors, and the connected red dots represent the neurons. The size of the network is determined before the training begins. As the map converges to represent the input data, there are some dead neurons floating between different data regions. [55]

2.3.2 Growing Self-Organizing Maps

The disadvantages of SOMs lead to the development of dynamically growing-SOMs (GSOMs) [11]. The training of GSOMs starts out with an initialization of usually four neurons with randomly initialized weights. Training occurs in the same manner as SOMs, but an accumulated error is calculated for each of the neurons to determine where new neurons should be inserted. These methods commonly worked by filling in neurons at every available space around the candidate neuron. This sometimes leads to the excessive growth of neurons, dead neurons with no associated input data, and the mis-configuration of the map. Mis-configuration occurs through the twisting of the neurons, where similar input data point would have very dissimilar neurons. For creating an IDS, GSOMs have been used as their growing phase can adapt to new attack types and provide better detection accuracy and false positive rates [39]. In [55], a method for perserving the topology

of the input data called directed batch GSOM (DBGSOM) was proposed. This method added growing rules to limit the growth of neurons in proper directions around a candidate neuron and initializing the weights of the added neurons to be similar to adjacent neurons. Figure 2.2 shows the twisting effect that occurs with GSOMs and how the DBGSOM better preserves topology of the input data. While the research and progress made on GSOMs achieved strides modeling data, the complexity and hierarchies of modern network data and the variety of modern attacks requires a more complex, hierarchical structure.

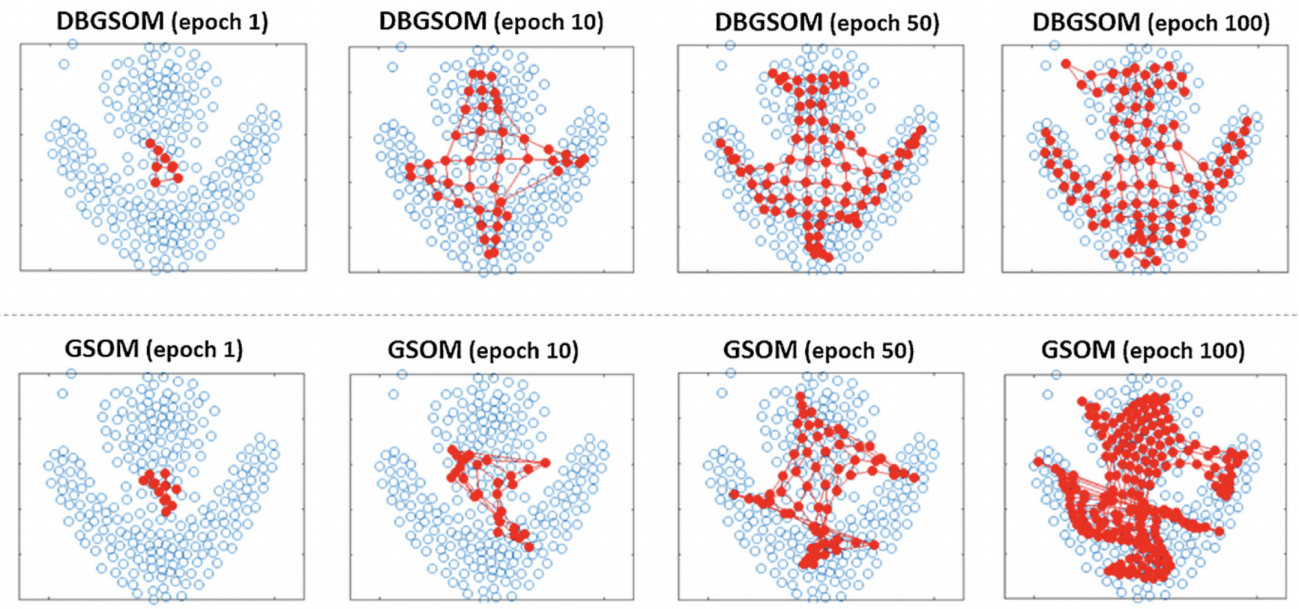


Figure 2.2: The bottom row shows the training process of GSOMs over a course of epochs. As neurons are added, misconfigurations and excessive growth of neurons results in poor topology preservation. DBGSOM has more limited growth criteria and neuron insertion rules resulting in better representation of the data and less useless neurons. [55]

2.3.3 Growing Hierarchical Self-Organizing Maps

The Growing Hierarchical Self-Organizing Map (GHSOM) captures this complexity through a vertical growth process [10]. Figure 2.3 shows the hierarchical structure of a GHSOM. During

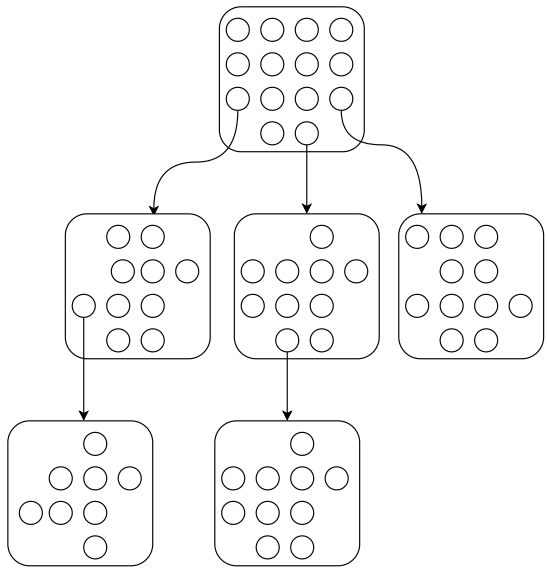


Figure 2.3: A simple illustration of the GHSOM architecture. The top of the hierarchy represents the root GSOMs and the circles represent neurons within. Neurons pointing to other GSOMs represent neurons with accumulated error greater than a certain threshold branching out to further model the data.

the modeling phase, a root GSOM is trained and the neurons of that map are iterated through. If the neuron has an accumulated error greater than a growth threshold parameter that node will branch off and become the parent of a child GSOM. That child GSOM will then be trained using the input data vectors where the parent neuron was the bmu. This process occurs recursively until there a no neurons with a great enough accumulated error. GHSOMs are able to model complex training data by breaking it down and modeling subsets of the data in lower layers. The literature on GHSOM, however, has been focused mainly on enhancing the algorithm and improving detection accuracy without taking explainability into account. For instance, the directed batch GHSOM (DBGHSOM) uses DBGSOMs for each node in the tree and provides high detection rates and low false positive rates, but the visualizations and explanations that made single layer SOMs appealing are absent [44]. An adaptive GHSOM (A-GHSOM) is a implementation that could accurately

predict unknown attacks through online adaptation of the model, but again, model interpretability was not available [16].

2.4 Pruning

One problem with GHSOMs is the difficulty in knowing when to stop the training process because a branch in a future subtree could provide critical information to the model. So models that grow very large and complex provide good detection rates but at the same time become harder to visualize and explain. Previously, Decision Trees faced a similar problem of being too large to gain knowledge from, so researchers sought to developed methods that simplifying the decision tree while retaining classification accuracy [45]. Decision Trees consist of nodes and branches, where at each node a feature value of the training data is observed in order to split the decision making process down separate pathways. When the observed data point reaches a terminal node, a prediction is made using the category of that node [37]. Figure 2.4 shows an example of what a

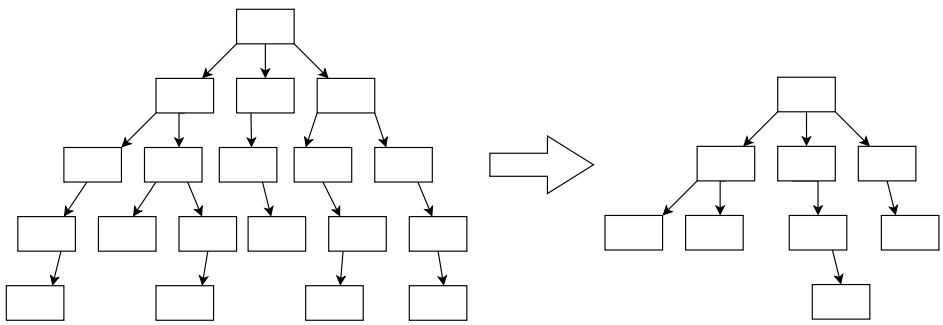


Figure 2.4: The left decision tree represents a decision tree with no pruning techniques applied. The unpruned tree can pose a difficult challenge of understanding all the criteria leading to a decision. The pruned tree on the right is easier to understand and less overfitted to the training data.

large tree may look like and how pruning improves the explainability of the model. Many different pruning methods were created and compared during this period. Error-Complexity pruning is a method that generates pruned trees from the unpruned tree and examines and selects a pruned tree by a measure of classification error and size of the tree [5]. Critical value pruning is a pruning method that occurs during the training process of a tree, where at each node during the training process, a critical value is assigned to the node based on how well that node splits up the classes of the data. If the critical value doesn't exceed a certain threshold then the node and its child nodes are pruned [34]. Pessimistic pruning algorithms work in a single pass through a tree, pruning nodes or subtrees where the removal does not reduce the training error significantly. This works on the idea that aggressively grown decision trees are overly optimistic that the model is not overfitted, and pessimistically removes branches would result in less overfitting [29]. Many of these methods have a common thread of using a measure of training error and a penalty based on the complexity of the tree to make the decision to prune a node. And though there has not been research exploring the use of pruning algorithms to improve the explainability of GHSOMs, using training error and a complexity penalty to prune GHSOMs in the same way is a natural extension.

CHAPTER 3

DATASETS

In this work, NSL-KDD [53] and CIC-IDS-2017 [41] were used to test the explainability and effectiveness of our architecture. NSL-KDD was chosen because of its wide use in the literature. The dataset is a improved version of its 1999 counterpart KDD'99 which was created in the Knowledge Discovery and Data Mining competition. The updated dataset removed many of the duplicate entries which helps reduce biases and improved the testing dataset to be more representative of real-world traffic. There are a few major benefits for using this dataset. First, it allows comparison to other existing IDS. Additionally, its relatively small size allows for quick testing and runtime comparisons against larger datasets. On the other hand, CIC-IDS-2017 includes more modern attacks and is useful for testing an unbalanced dataset. Many datasets, at the time of CIC-IDS-2017's creation, were out of date and not representative of current network data. It was synthetically created over the course of 5 days to mimic the behavior of 25 users. Using this dataset shows that the proposed IDS is useful when trained with real-world data. The preprocessing of the data includes feature selection and normalization. The feature selection algorithm is a Bayesian Probability of Significance [13] that select the most relevant features from each dataset. The only other preprocessing that was performed on these datasets was normalization. After preprocessing

is finished, the new, high-quality dataset can then be passed to the model. The next section details information about the previously mentioned datasets and their usefulness in testing IDS.

CHAPTER 4

METHODOLOGY

This section will layout the proposed methodology for the experiments. First, the DBGSOM and DBGHSOM algorithm that will be used, the visualizations garnered from the models, and finally the pruning algorithm that will be used.

4.1 DBGSOM

The training of a DBGSOM, described in [55], consists of first initializing a square grid of neurons with random weights and defining an amount of training epochs. A horizontal growing threshold, HGT is then also set:

$$HGT = -D(\ln(SF)) \tag{4.1}$$

Where D is the dimensionality of the input data and SF is the spreading factor between the value of 0 and 1. Lower SF values limit the growth of the SOM and higher values increase it.

After initialization, the growing phase starts. At the start of each epoch, the accumulated error of every neuron is set to 0. Every input data vector is then compared, using Euclidean distance, to each neuron weight vector and assigned a winning neuron closest to it. Next the neuron weights are updated. For every neuron, the data points assigned to that neuron are averaged and weighted by:

$$w_i^{new} = \frac{\sum_{j=1}^{k} h_{cj,i} x_j}{\sum_{j=1}^{k} h_{cj,i}} \tag{4.2}$$

Where w_i^{new} is the the new weights for the i^{th} neuron and x_j is the j^{th} data vector assigned to that neuron. The Voronoi set for a winning neuron are the k data vectors that all have that same winning neuron. And $h_{cj,i}$ is the Gaussian neighborhood function:

$$h_{cj,i} = \exp\left(-\frac{||w_i - w_{cj}||^2}{2\sigma^2(t)}\right) \tag{4.3}$$

Where w_i is the average of all the data vectors assigned to the i^{th} neuron and w_{cj} is the weight vector of the j^{th} input data vector of the winning neuron. And σ is a neighborhood bandwidth parameter that decays as the number of nodes increases.

Once the weights of all the neurons are updated, the accumulated error, e_i, is calculated for each neuron by taking a summation of the distances between the weights of the winning neuron, w_i, and the data vectors in its Voronoi set, x_j:

$$e_i = \sum_{j=1}^{k} ||x_j - w_i|| \tag{4.4}$$

For non-boundary neurons, i.e neurons that have no free adjacent space to insert a neuron, the accumulated error of that neuron is divided in half and distributed evenly among its neighbors. For boundary neurons whose accumulated error surpasses the growth threshold in equation 4.1, different rules are specified in [55] for neuron insertion and weight initialization. When deciding to insert a neuron where the boundary neuron has 2 or more open positions, a neuron is inserted in the area where the accumulated error for the neuron neighboring the open position is greatest.

When a neuron is inserted in a position that has an adjacent neuron, the weight initialization of the inserted neuron is:

$$w_{new} = \frac{(2w_b - w_{nb}) + w_{nb}(i)}{2} \tag{4.5}$$

Where w_b is the weight of the boundary neuron, W_{nb} is the weight of the neuron with the highest accumulated error neighboring the boundary neuron, and $w_{nb}(i)$ is the weight of the neuron neighboring the available position. When there are no neurons neighboring the available position the weight of the inserted neuron is simply:

$$w_{new} = 2w_b - w_{nb} \tag{4.6}$$

Once all the neurons and inserted and their weights initialized, the training epoch is over and the process is repeated until the specified number of epochs is reached. Labels are then assigned to neurons based on the most prevalent class mapped to the neurons.

4.2 DBGHSOM

The training process DBGHSOM starts with training a DBGSOM on all of the input training data. This will be the root node of the tree and, once it is trained, a vertical growing process occurs [42]. First the vertical growth threshold, VGT, to determine vertical growth is calculated by multiplying a learning parameter, λ, by the sum of accumulated errors of the DBGSOM:

$$VGT = \lambda \sum_{k=1}^{m} e_k \tag{4.7}$$

Then every neuron's individual accumulated error is compared with VGT, and if VGT is greater, then a DBGSOM is trained using the Voronoi set for that neuron. Every new DBGSOM will undergo this vertical growth until no neurons of any DBGSOM surpass the vertical growth threshold.

4.3 Pruning

A pessimistic pruning method based on [18, 29] will be used on the fully trained DBGHSOM. During one bottom-up pass through the tree, a decision is made at every node to keep a DBGSOM or delete it and its possible children. For a node that is not a leaf node, every child from the subtree rooted at the node needs to have gone through the decision to keep or prune before a decision can be made for a subtree. The pruning decision is based on a comparison in equation 4.8 of the error rate for the best leaf of a subtree, e_{bl}, and the training error rate of the subtree, e_{st}, plus a tree complexity penalty, α. All error rates are based on the local input data points mapped to the subtree:

$$e_{st} + \alpha \geq e_{bl} \tag{4.8}$$

$$\alpha = c\sqrt{\frac{(l+s)\log(n) + \log(\frac{m}{\delta})}{m_v}} \tag{4.9}$$

For the complexity penalty at any node in the tree, l is the depth of that node, s is the number of nodes in the subtree, n is the total number of nodes in the tree, m is total size of the training data, m_l is the local size of the data mapped to the subtree, δ is a confidence parameter, and some constant c used to control the amount of pruning.

4.4 Visualizations

Once trained, GHSOMs illustrate mappings between data points and the associated BMU. As this is generally a 2D representation of the feature space, it can be visually understood by the user. GHSOM's explainablity can be divided into *Global* and *Local* explanations.

Global explanations are used to give a general idea of how a particular model computes predictions. The U-Matrix , is a commonly used technique [54]. This additive metric works by summing the distance to each of a unit's neighbors. A group of low scores will represent clusters in the map, while a group of high scores will signifies sparseness. For more fine-grained data, feature heat-maps will be created to visualize GHSOM feature values and their importance. Another visualization method called a tree-map will visualize the size and class labels of neurons in the DBGHSOM. These techniques will provide global explanations.

Local explanations will be generated for individual sample datapoints and will be used to explain why a certain prediction value was computed. This allows the user to understand the decision process of the SOM model. The primary use of this method is to explain and visualize feature importance. When making a prediction, a datapoint's features will be scored based on how impactful they are to the computed prediction. Wickramasinghe et al. [56] developed an explainable SOM for Cyber-Physical Systems. Their system created both local and global explanations by data-mining a SOM model. The mined information was used to create visualizations including histograms, T-distributed Stochastic Neighbor Embedding (t-SNE), heat maps, U-Matrix, component planes, and U-Map. This variety of visualizations allow the SOM to be explainable not only to domain experts, but also non-domain experts. Extending this to GHSOMs and creating visualizations for

the hierarchical structure will better represent more complex data, explain decisions, and remain accurate.

CHAPTER 5

RESULTS

5.1 Experiment Results

The experimental results will consist of accuracy, precision, recall, f1, false positive rate, false negative rate, and network size measures. Accuracy refers to the percentage of correct predictions compared to the total test size. Precision measures the ratio of true positive predictions to the total number of positive predictions. Recall is the measure of true positives predictions to the total number of positive samples in the test set. The f1 score is a measure that gives equal weight of precision and recall. False positive rate is the rate of false positive predictions compared to the amount of ground truth negatives. False negative rate is the rate of false negative predictions compared to the amount of ground truth positives. And finally, network size is simply the amount of GSOMs within the hierarchical GHSOM structure. For SOMs and GSOMs the network size will just be 1. All results can be found in Table 5.1

5.1.1 SOM Results

The SOM training process consists of 1000 iterations of training on an 18x18 neuron map. For NSL-KDD, SOM obtains an accuracy of 90.7%, precision of 97.2%, recall of 83.3%, F1 of 89.7%, FPR of 2.2%, and a FNR of 16.6%. The training time takes 8 seconds with an average inference time of .03 milliseconds. On CIC-IDS, the model achieves a 79.4% accuracy, 83.2% precision,

Table 5.1: Prediction results for SOM, GSOM, GHSOM, and Pruned GHSOM

Dataset	Measure	SOM	GSOM	GHSOM	Pruned GHSOM
nslkdd100					
	Accuracy	90.9%	96.0%	98.2%	98.0%
	Precision	97.2%	96.9%	98.0%	98.0%
	Recall	83.3%	94.7%	98.3%	97.8%
	F1	89.7%	95.8%	98.1%	97.9
	FPR	2.2%	2.9%	1.9%	1.8%
	FNR	16.6%	5.2%	1.6%	2.2%
	Network Size	1	1	7288	574
	Training Time (s)	8	60	693	816
	Avg. Prediction Time (ms)	.03	.03	.06	.04
cicids2017					
	Accuracy	79.4%	95.1%	96.7%	95.7%
	Precision	83.2%	84.8%	89.1%	86.5%
	Recall	42.0%	91.3%	94.5%	92.7%
	F1	55.8%	87.9%	91.7%	89.5%
	FPR	19.0%	4.0%	2.8%	3.5%
	FNR	23.0%	8.7%	5.5%	7.3%
	Network Size	1	1	16894	119
	Training Time (s)	260	768	4299	11205
	Avg. Prediction Time (ms)	.03	.03	1.15	.03

42.0% recall, 55.8% F1 score, 19.0% FPR, and a 23.0% FNR. Due to the large amount of training data the training time is longer at 260 seconds, but since the size of the map remained the same, the average prediction time is still .03 seconds.

5.1.2 GSOM Results

For NSL-KDD, the DBGSOM was trained for 100 epochs with a spreading factor of .9 causing an aggressive growth of neurons and quicker topological convergence. The model achieves an accuracy of accuracy of 96.0%, precision of 96.9%, recall of 94.7%, F1 of 95.8%, FPR of 2.9%, and a FNR of 5.2%. It takes 60 seconds to train and .03 ms to make a prediction. CIC-IDS was trained over 40 epochs and a spreading factor of .9 and resulted in a 95.1% accuracy, 84.8% precision, 91.3% recall, 87.9% F1 score, 4.0% FPR, and a 8.7% FNR. The model took 768 seconds to train and can make an inference every .03 ms.

5.1.3 GHSOM Results

The DBGHSOM for NSL-KDD was again trained for 100 epochs but at a lower spreading factor of .3. This causes the root map to grow less and allows for more of the hierarchies of the input data to be captured in lower layers and branches of the the tree. This model obtained a 98.2% accuracy, 98.0% precision, 98.3% recall, 98.1% F1 score, 1.9% FPR, and a 1.6% FNR. The network size was 7288 total GSOMs and took 693 seconds to train. The average prediction time is slightly higher at .06 ms due to the larger network size. Pruning this model with a δ of 10 and a c value of .3 reduced the size of the network to 574 and maintained an accuracy of 98.0%, precision of 98.0%, recall of 97.8%, F1 of 97.9%, FPR of 1.8%, and a FNR of 2.2%. The time to prune the

network results in a higher training time of 816 seconds, but the reduced network size also reduced the average prediction time to .04.

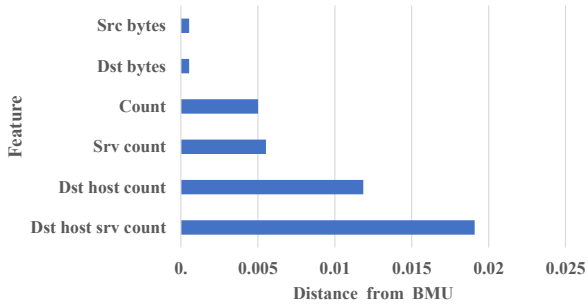
The DBGHSOM trained on CIC-IDS also has a lower spreading factor of .3 to allow for more vertical growth rather than horizontal growth. It was trained over 40 epochs and resulted in an accuracy of 96.7%, precision of 89.1%, recall of 94.5%, F1 of 91.7%, FPR of 2.8%, FNR of 5.5%, and a network size of 16894. DBGHSOM had the a training time of 4299 seconds with the highest average prediction time of 1.15 ms. Pruning with the same parameters used to prune the NSL-KDD model reduced the network size to 119. This model achieved an accuracy of 95.7%, precision of 86.5%, recall of 92.7%, F1 of 89.5%, FPR of 3.5%, FNR of 7.3%. The training time of 11205 seconds was the longest, but it reduced the average prediction time to .03 ms.

5.2 Visualization and Explanation

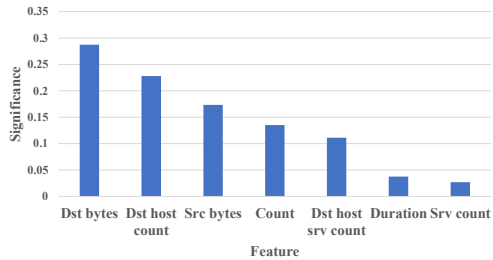
This section will discuss the results from each of the visual and graphical explanations for all models.

5.2.1 SOM Explanations

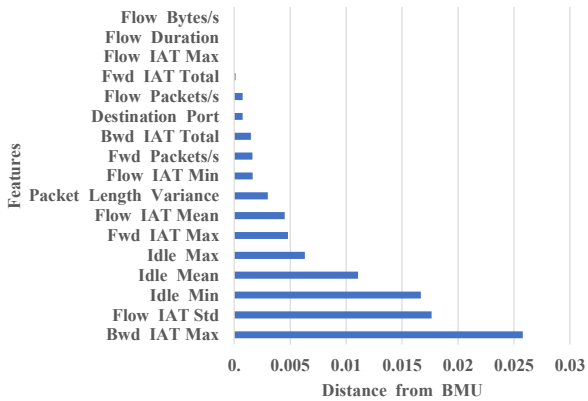
The results for the NSL-KDD dataset can be found in Figures 5.1a and 5.1b. The local explanation example shows that the most important features for its prediction were 'Duration', 'Destination (dst) bytes', and 'Source (src) bytes'. The remaining features, 'Service (srv) count', 'Count', and 'Destination (dst) host count' are considered less significant because of their distance from the BMU. Two of the important features coincide with the Global Feature Significance graph. This trend continues when testing on many different test samples. The most important global features are frequently at the forefront for local significance. Similarly to NSL-KDD, CIC-IDS-



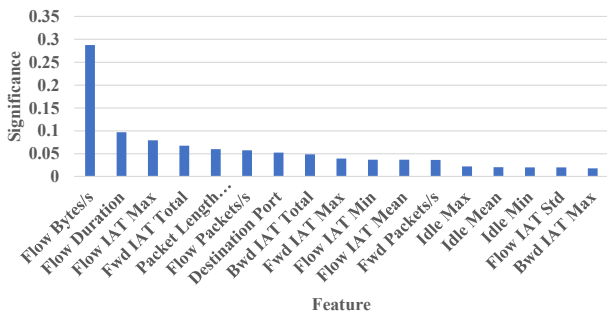
(a) NSL-KDD Local Anomaly Explanation



(b) NSL-KDD Global Feature Significance

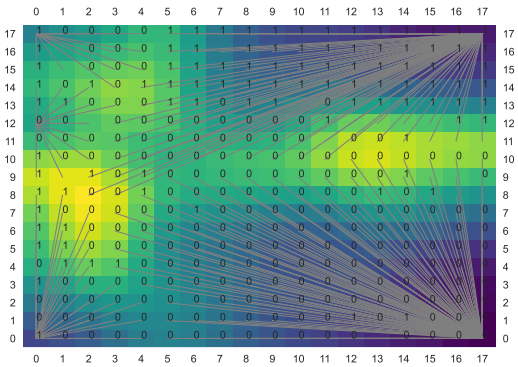


(c) CIC-IDS-2017 Local Normal Explanation

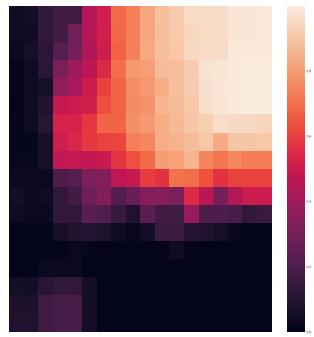


(d) CIC-IDS-2017 Global Feature Significance

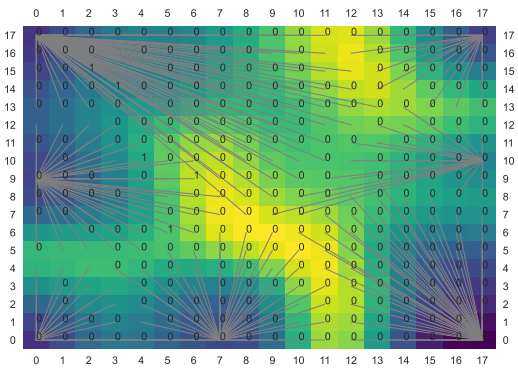
Figure 5.1: These figures show the local and global feature explanations for both the NSL-KDD and CIC-IDS datasets. (a) The local explainability of a prediction is defined by the distance between feature value and BMU. More important features have a lower score than less important features. This figure shows the feature importance for an anomalous sample from the NSL-KDD testing set. (b) Global feature significance is calculated using Bayesian Probability of Significance [13]. Higher values are considered more important than lower values.



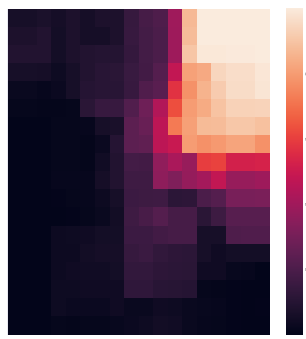
(a) NSL-KDD Starburst U-Matrix



(b) Dst byte Feature Map



(c) CIC-IDS-2017 Starburst U-Matrix



(d) Flow bytes/s Feature Map

Figure 5.2: (a)(d) The Starburst U-Matrix shows both the most common label for each node and the clusters the SOM has learned. Darker areas represent units that are close Euclidean Distance-wise. Notably, we can see a clear divide between classes on the NSL-KDD dataset as represented in the figure. In this model's iteration, anomalous traffic is mostly grouped on the top of the SOM.(c) The feature value heatmap displays the value of a specific feature on each unit in the SOM. Lighter values represent units with values closer to 1, while darker values show values closer to 0.

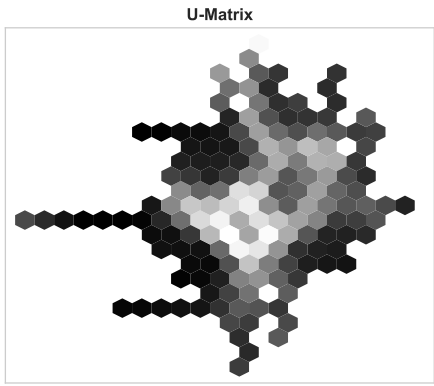
2017 follows this trend. Many of the top, globally selected features also play a more important role in the local predictions.

The next set of explainability techniques has been data-mined from the trained SOM. Figures 5.2a and 5.2c show the generated Starburst U-Matrix for NSL-KDD and CIC-IDS-2017, respectively. The SOM algorithm was able to separate benign clusters from malicious clusters in the map created from NSL-KDD dataset. The top-right corner is primarily malicious samples, while the bottom-right corner contains mostly benign samples. Additionally, the clusters marked by the starbursts' origins mostly represent one label. On the other hand, the CIC-IDS-2017 map has not separated the labels sufficiently. Most of the labels present in the figure are benign (0) with very few malicious labels (1).

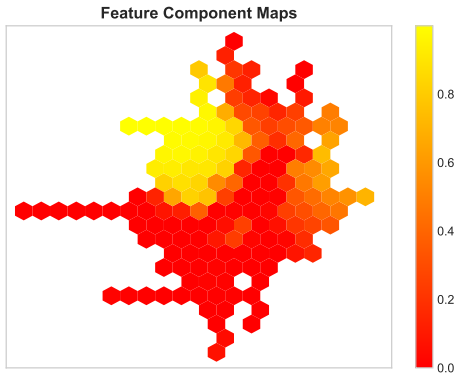
Lastly, the feature value heatmaps are generated for each feature of the dataset. The examples chosen were the most significant features for each dataset: 'destination (dst) bytes' and 'flow bytes/s'. On their own, they can be used to see general training trends for each feature. In Figures 5.2b and 5.2d, we can see that each of these features have higher values in the top-right units and lower values elsewhere.

5.2.2 GSOM Explanations

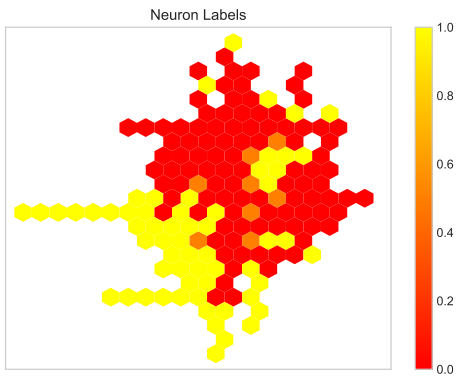
The visualizations and explanations for GSOM are very similar to the basic SOM. Local explanations can still gained from predictions, and the global explanation for the model will remain the same as it is with SOM. Figure 5.3 shows the visualizations from a DBGSOM trained on the NSL-KDD dataset. The U-matrix and feature component map are shown in Figure 5.4a and Figure 5.4b, respectively. A noticeable difference is the shape of the figures being irregular compared



(a) U-Matrix

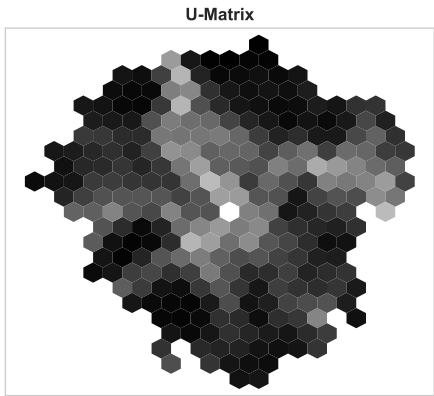


(b) Feature Component Map of Dst-Bytes

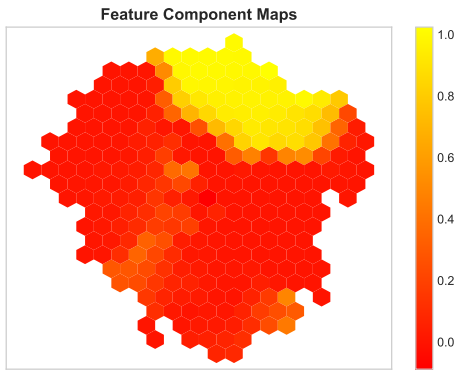


(c) Neuron Labels

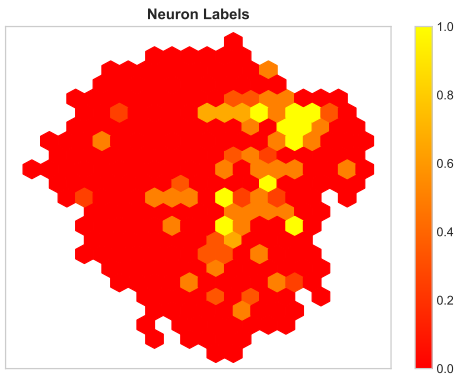
Figure 5.3: Visualizations generated from a DBGSOM trained on NSL-KDD. (a) The U-matrix maintains the same properties as the SOM starburst visualization with darker areas representing neurons closer together. (b) The Feature Component Map also shares the same properties to the SOM feature map in Figure 5.2. (c) The Neuron Label map shows the class label represented by a red or yellow color.



(a) U-Matrix



(b) Feature Component Map of Flow Bytes/s



(c) Neuron Labels

Figure 5.4: Visualizations generated from a DBGSOM trained on CIC-IDS-2017. (a)(b)(c) Share the same properties with Figure 5.3

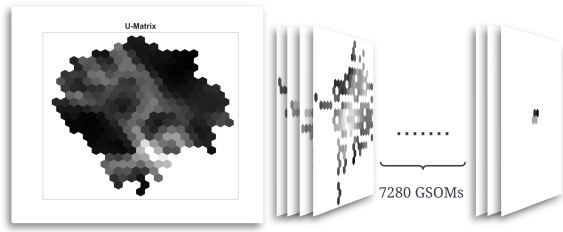
to SOM due to the fact that DBGSOMs grow in the direction towards the most error. Another difference is the the way neurons are depicted, as hexagons rather than squares, to better represent the irregular structure. Due to these differences, the starburst model used for SOM is replaced by a U-matrix and a neuron label map. The label map, depicted in Figure 5.4c, has the color of each hexagon representing the class label of that neuron. Figure 5.4 depicts the visualizations generated from DBGSOM after training on the CIC-IDS-2017 dataset.

5.2.3 GHSOM Explanations

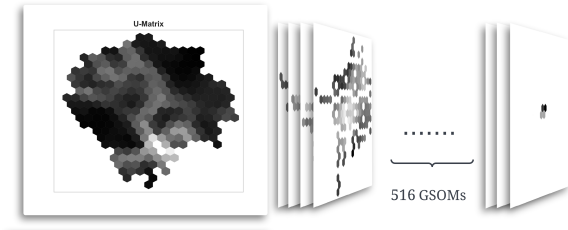
As DBGHSOMs are comprised of a hierarchy of many different DBGSOMs, the visualizations are also more complex. Figure 5.5 shows stacked depictions of unpruned and pruned U-matrices (Figure 5.5a, 5.5b), feature component maps (Figure 5.5c, 5.5d), and neuron label maps (Figure 5.5e, 5.5f) that are generated from the DBGSOMs that form the hierarchy. The unslanted figures on the lefthand side of each figure represents the root node of the DBGHSOM. Going from left to right in the figures shows DBGSOMs deeper in the tree's hierarchy. The ellipsis and number below it in the middle of each figure represents the amount of figures not shown. The maps become more sparse in deeper layers of the hierarchy and the leaf nodes tend to have four neurons which is depicted on the right hand side of each figure.

Another method for visualizing a trained GHSOM is the tree-map, which is depicted in Figure 5.6. This is made using the CIC-IDS dataset without utilizing the pruning method. Each larger box in the picture represents a DBGSOM with a label serving as an identifier and the layer at which it resides. Smaller boxes within the GSOM represents neurons within the GSOM and are labeled with an identifier and its size; the physical size of each neuron is based on the number of samples

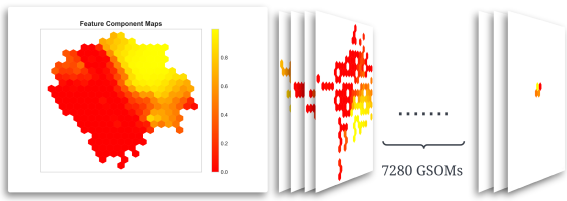
where that neuron is the bmu. The colors of each neuron represents the label of the neuron, unless the neuron has a child DBGSOM, in which case the identifier of the child DBGSOM is displayed on the neurons. Figure 5.7 displays the resulting tree-map of the model shown in Figure 5.6 after undergoing the pruning process.



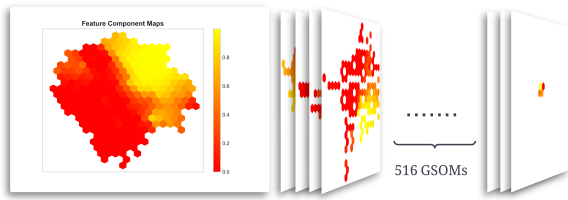
(a) U-matrices



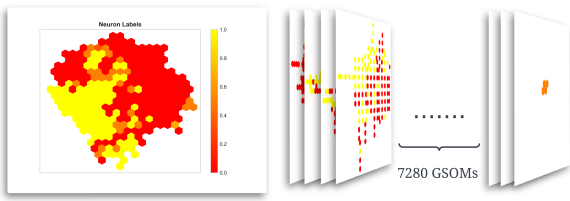
(b) Pruned U-matrices



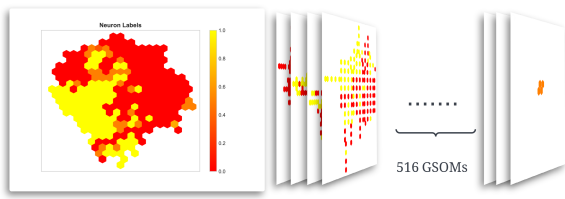
(c) Feature Maps



(d) Pruned Feature Maps



(e) Label Maps



(f) Pruned Label Maps

Figure 5.5: Visualizations created from GHSOM trained on NSL-KDD. The left hand unslanted visualization represents the root node. The slanted visualizations represent the DBGSOMs deeper in the hierarchy of DBGHSOM. Figures (a), (c), and (e) show an unpruned DBGHSOM with 7280 unseen DBGSOMs, while Figures (b), (d), and (f) show the pruned version with 516 unseen DBGSOM visualizations.

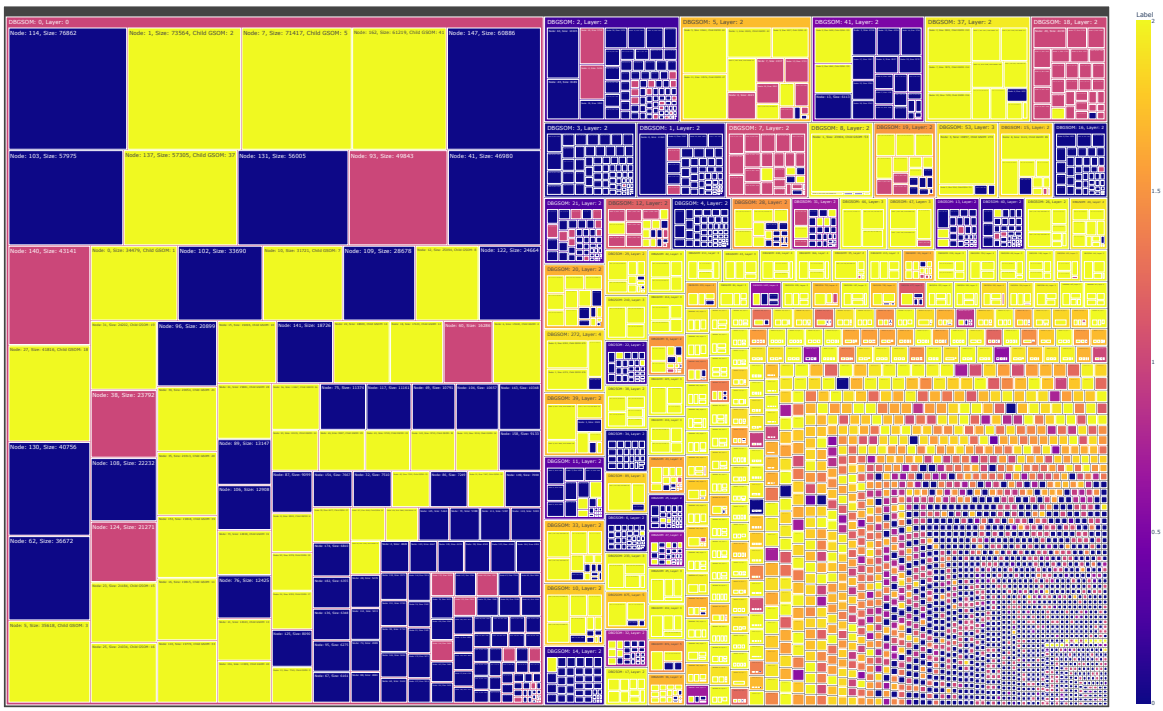


Figure 5.6: A tree map containing the results of training a DBGHSOM on the CIC-IDS-2017 dataset. Larger boxes are DBGSOMs that are made up of neurons with each DBGSOM and neuron being labeled with an identifier. Red and blue nodes represent malicious and benign data, respectively. Yellow nodes represent a branch occurring at that node and show the identifier of its child DBGSOM.

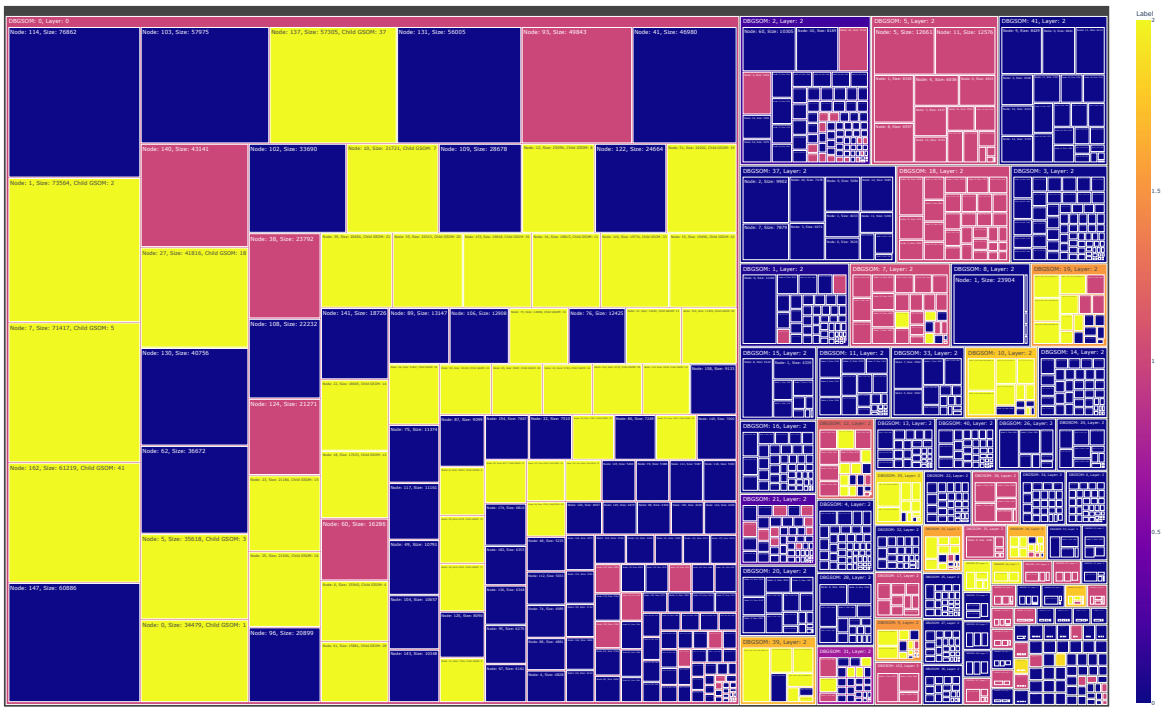


Figure 5.7: The tree map generated after pruning the DBGHSOM from Figure 5.6

CHAPTER 6

DISCUSSION

6.1 Model Performance

Similar trends between the NSL-KDD models and CIC-IDS models can be seen when comparing results from Table 5.1. SOMs tend to perform the worst with a severe drop and accuracy and a increased false positive and negative rate for both NSL-KDD and CIC-IDS. The DBGSOM models offer competitive performance to the hierarchical models but with slightly less performance but with the benefit of containing all of the information in one self contained map. GHSOMs provide the highest performance around the board with decreased false positive rates and false negative rates along with higher accuracy measures. They do have a very high network size however with the models trained on both datasets having thousands DBGSOM nodes. The pruned GHSOM for NSL-KDD has only a slight decrease in accuracy measures and slight increases in classification rates while also reducing the network size significantly from 7288 nodes to 574 nodes. The decrease in accuracy for the pruned model trained on CIC-IDS-2017 is greater, but the reduction in network size was much greater, dropping it from 16894 nodes to 119 nodes. A less aggressive pruning process may have maintained the accuracy of the model while still reducing the network size by a significant amount.

6.2 Model Explainability

For all models, Figure 5.1 shows the local explanations that are made for predictions and the global importance variables have when creating the models. The following subsections will discuss the unique visualizations generated from each model, the quality of explanations they produce, and how they can be used with the local and global explanations to gain understanding about the input data and predictions made.

6.2.1 SOM

When analyzing the starburst matrix in Figure 5.2a and Figure 5.2c, each neuron is labeled with the class label used when making predictions. Observing the neurons close to each other, denoted as darker areas in the map, along with class labels gives information about areas of the data that are closely associated with malicious network behavior. For NSL-KDD, the top right corner of the map is an area of interest because all of that areas neurons are labeled as malicious. Used in conjunction with global and local explanations, important features such as dst bytes, which is the number of bytes going from the destination to the source connection, can be observed in Figure 5.2b. This comparison shows a correlation with high dst bytes values and the neuron cluster with malicious labels. One can go further looking at other important feature component maps to gain a more complete picture of why the model came to a certain prediction. Users will be able to build a mental model of the SOM when visualized in conjunction with the features maps. For example, if ‘destination (dst) byte’, ‘duration’, and ‘source (src) byte’ all have higher values in the malicious section of the map then One may conclude that when these values are all close to one, the sample is more than likely malicious.

The same kind of explanations can be gained from the SOM model trained on CIC-IDS. In this case, however, the SOM algorithm is inadequate in modeling the input data and does not provide high quality visualizations. Nearly all neurons are labeled as benign with a small amount of neurons labeled as malicious spread throughout the map. CIC-IDS-2017 is an unbalanced dataset, with about 70% of samples being benign and 30% of samples as malicious. This class imbalance causes the SOM label neurons as benign rather than malicious.

6.2.2 GSOM

Visualizations and explanations garnered from DBGSOMs are very similar. But due to the irregular nature of the models, the starburst model from SOM is replaced with a U-matrix and label map. The label map in this case does appear easier to read and compare to the other maps as the neurons are brightly colored to denote class while the class labels on the starburst U-matrix blend in and are harder to read. Figure 5.3 shows the visualizations created from training on NSL-KDD. Comparing the label map in Figure 5.4c and the feature component map in Figure 5.4b shows an interesting deviation from the explanations generated from the SOM visualizations. Here low dst-byte values correlate to malicious network traffic, the opposite of the conclusion drawn from the SOM model. Here the DBGSOM saw a higher potential in low dst-byte values and ended up being more trustworthy with higher accuracy scores than the SOM model.

Figure 5.4 shows the visualizations created for a DBGSOM trained on the CIC-IDS-2017 dataset. One of the important features for this dataset is flow bytes/s, which is the number of packets transferred per second. Comparing the feature map to the neuron label map shows that there are a group of neurons labeled as malicious that have high flow bytes/s neuron weights.

However, There are also many neurons labeled as malicious with low flow bytes/s neuron weights. With the variety of modern attack types and profiles present within the CIC-IDS dataset, the more complex model visualizations of GHSOM can help a security analyst understand more complex data.

6.2.3 GHSOM

The visualizations created for DBGHSOM are inherently the same, but a single model is made up of a hierarchy of multiple DBGSOM visualizations. Figure 5.5 shows the stacked U-matrices, feature maps, and label maps. DBGHSOM provides a highly accurate model at the cost of the simplicity of the model, though this may be preferable to a security analyst who is exploring the intricacies of a complex dataset. To gain a local explanation, a security analyst can dig into the hierarchy to the DBGSOM that made the classification and explore the properties of those visualizations. While this maybe useful to find different hierarchies of the data, a DBGHSOM grown too large become to complex for anyone to understand. Figure 5.6 shows a tree-map of a model trained on CIC-IDS, and while making sense of the root and upper layers may be feasible, the lower layers and branches are so numerous that a cogent analysis of the model would be impossible.

Pruning of the DBGHSOM maintains a model which captures the complexity of the data, is highly accurate, and can be easily explored. Figure 5.7 demonstrates the tree-map from Figure 5.6 after undergoing pruning. Compared to the unpruned tree-map, there are many more visible DBGSOMs that can be explored by visualizing the size and classes of the neurons in each DBGSOM and by using the DBGSOM identifiers to analyze the visualizations from Figures 5.5b, 5.5d, and 5.5f. There are DBGSOMs with mostly benign and a with a few malicious neurons that a

security analyst can use to understand network traffic that is close to being benign but is actually malicious. On the other hand, there are some DBGSOMs in the tree-map that consist of mostly malicious neurons with a few benign neurons. This could be analyzed to flag possible false negative predictions as a possible threat. DBGSOMs that are entirely malicious or benign can be used to understand the intricacies of different types of malicious and benign behavior.

CHAPTER 7

CONCLUSION

This paper demonstrated the capabilities of creating an explainable, highly accurate IDS. GH-SOMs were able to reach 98.2% and 96.7% accuracies on the NSL-KDD and CIC-IDS-2017 datasets, respectively. A visualization hierarchy and tree-map created a way to explore the complexities and hierarchies of the data and gain insights about why a prediction was made. A pruning method was also implemented to limit the size of the DBGHSOM, maintaining high accuracy levels and providing better visualizations and thus making insights easier to gain. A comparison with the performance and explainability showed DBGHSOM and the pruned DBGHSOM to perform better than the single layer SOM and DBGSOM, while also keeping the visualizations that make SOM an appealing choice for intrusion detection systems. Future work may include studying the effect on performance and network size by using different pruning methods or varying the parameters of the proposed pruning technique.

REFERENCES

[1] J. Ables, T. Kirby, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, and M. Seale, "Creating an Explainable Intrusion Detection System Using Self Organizing Maps," *ArXiv*, vol. abs/2207.07465, 2022.

[2] R. G. Bace, P. Mell, et al., "Intrusion detection systems,", 2001.

[3] M. Belouch, S. El Hadaj, and M. Idhammad, "Performance evaluation of intrusion detection based on machine learning using Apache Spark," *Procedia Computer Science*, vol. 127, 2018, pp. 1–6.

[4] A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, and W. Samek, "Layer-wise relevance propagation for neural networks with local renormalization layers," *International Conference on Artificial Neural Networks*. Springer, 2016, pp. 63–71.

[5] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, Routledge, 2017.

[6] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, 2009, pp. 15:1–15:58.

[7] Y. E. Chun, S. B. Kim, J. Y. Lee, and J. H. Woo, "Study on credit rating model using explainable AI," *The Korean Data & Information Science Society*, vol. 32, no. 2, 2021, pp. 283–295.

[8] DARPA, "Broad agency announcement explainable artificial intelligence (XAI)," *DARPA-BAA-16-53*, 2016, pp. 7–8.

[9] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, , no. 2, 1987, pp. 222–232.

[10] M. Dittenbach, D. Merkl, and A. Rauber, "The growing hierarchical self-organizing map," *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. IEEE, 2000, vol. 6, pp. 15–19.

[11] B. Fritzke, "Growing Grid — a self-organizing network with constant neighborhood range and adaptation strength," *Neural Processing Letters 1995 2:5*, vol. 2, 9 1995, pp. 9–13.

[12] D. Gunning and D. Aha, "DARPA's explainable artificial intelligence (XAI) program," *AI Magazine*, vol. 40, no. 2, 2019, pp. 44–58.

[13] L. Hamel and C. Brown, "Bayesian Probability Approach to Feature Significance for Infrared Spectra of Bacteria," *Applied Spectroscopy*, vol. 66, 1 2012, pp. 48–59.

[14] M. Han and J. Kim, "Joint banknote recognition and counterfeit detection using explainable artificial intelligence," *Sensors*, vol. 19, no. 16, 2019, p. 3607.

[15] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell, "What do we need to build explainable AI systems for the medical domain?," *arXiv preprint arXiv:1712.09923*, 2017.

[16] D. Ippoliti and X. Zhou, "A-GHSOM: An adaptive growing hierarchical self organizing map for network anomaly detection," *Journal of Parallel and Distributed Computing*, vol. 72, 12 2012, pp. 1576–1590.

[17] S. R. Islam, W. Eberle, S. K. Ghafoor, A. Siraj, and M. Rogers, "Domain knowledge aided explainable artificial intelligence for intrusion detection and response," *arXiv preprint arXiv:1911.09853*, 2019.

[18] M. Kearns and Y. Mansour, "A Fast, Bottom-Up Decision Tree Pruning Algorithm with Near-Optimal Generalization," *ICML*, 1998.

[19] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, 1982, pp. 59–69.

[20] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, 1998, pp. 1–6.

[21] C. Langin, M. Wainer, and S. Rahimi, "ANNaBell Island: a 3D color hexagonal SOM for visual intrusion detection," *Internation Journal of Computer Science and Information Security*, vol. 9, no. 1, 2011, pp. 1–7.

[22] K. Letou, D. Devi, and Y. Jayanta, "Host-based Intrusion Detection and Prevention System (HIDPS)," *International Journal of Computer Applications*, vol. 69, 05 2013, pp. 28–33.

[23] P. Lichodzijewski, A. N. Zincir-Heywood, and M. I. Heywood, "Host-based intrusion detection using self-organizing maps," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, 2002, pp. 1714–1719.

[24] L. Lindsay, S. Coleman, D. Kerr, B. Taylor, and A. Moorhead, "Explainable Artificial Intelligence for Falls Prediction," *International Conference on Advances in Computing and Data Sciences*. Springer, 2020, pp. 76–84.

[25] Z. C. Lipton, "The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery.," *Queue*, vol. 16, no. 3, 2018, pp. 31–57.

[26] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[27] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[28] B. Mahbooba, M. Timilsina, R. Sahal, and M. Serrano, "Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model," *Complexity*, vol. 2021, 2021.

[29] Y. Mansour, "Pessimistic decision tree pruning based on tree size," *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*. Citeseer, 1997, pp. 195–201.

[30] A. Marshan, "Artificial intelligence: Explainability, ethical issues and bias," *Annals of Robotics and Automation*, 08 2021, pp. 034–037.

[31] A. McDole, M. Abdelsalam, M. Gupta, and S. Mittal, "Analyzing CNN based behavioural malware detection techniques on cloud IaaS," *International Conference on Cloud Computing*. Springer, 2020, pp. 64–79.

[32] A. McDole, M. Gupta, M. Abdelsalam, S. Mittal, and M. Alazab, "Deep Learning Techniques for Behavioural Malware Analysis in Cloud IaaS," *Malware Analysis using Artificial Intelligence and Deep Learning*, Springer, 2021.

[33] S. McElwee and J. Cannady, "Improving the performance of self-organizing maps for intrusion detection," *Conference Proceedings - IEEE SOUTHEASTCON*, vol. 2016-July, 7 2016.

[34] J. Mingers, "Expert Systems—Rule Induction with Statistical Data," *Journal of the Operational Research Society 1987 38:1*, vol. 38, 1 1987, pp. 39–47.

[35] J. D. Moore and W. R. Swartout, *Explanation in expert systemss: A survey*, Tech. Rep., University of Southern California Marina Del Rey Information Sciences Inst, 1988.

[36] B. Mukherjee, T. L. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Network*, vol. 8, 1994, pp. 26–41.

[37] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics*, vol. 18, 6 2004, pp. 275–285.

[38] S. Neupane, J. Ables, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, and M. Seale, "Explainable Intrusion Detection Systems (X-IDS): A Survey of Current Methods, Challenges, and Opportunities,", 2022.

[39] E. J. Palomo, E. Domínguez, R. M. Luque, and J. Muñoz, "A self-organized multiagent system for intrusion detection," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5680 LNAI, 2009, pp. 84–94.

[40] G. Pang, C. Ding, C. Shen, and A. v. d. Hengel, "Explainable Deep Few-shot Anomaly Detection with Deviation Networks," *arXiv preprint arXiv:2108.00462*, 2021.

[41] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems," *International Journal of Engineering & Technology*, vol. 7, 3 2018, pp. 479–482.

[42] X. Qu, L. Yang, K. Guo, L. Ma, T. Feng, S. Ren, and M. Sun, "Statistics-enhanced direct batch growth self-organizing mapping for efficient dos attack detection," *IEEE Access*, vol. 7, 2019, pp. 78434–78441.

[43] X. Qu, L. Yang, K. Guo, L. Ma, M. Sun, M. Ke, and M. Li, "A Survey on the Development of Self-Organizing Maps for Unsupervised Intrusion Detection," *Mobile Networks and Applications 2019 26:2*, vol. 26, 10 2019, pp. 808–829.

[44] X. Qu, L. Yang, K. Guo, M. Sun, L. Ma, T. Feng, S. Ren, K. Li, and X. Ma, "Direct Batch Growth Hierarchical Self-Organizing Mapping Based on Statistics for Efficient Network Intrusion Detection," *IEEE Access*, vol. PP, 02 2020, pp. 1–1.

[45] J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-Machine Studies*, vol. 27, 1987, pp. 221–234.

[46] Raytheon, "Cyber Security Operations Center (CSOC),", 2017.

[47] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?" Explaining the predictions of any classifier," *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[48] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. C. Platt, "Support Vector Method for Novelty Detection," *NIPS*, 1999.

[49] A. Sharma and S. K. Sahay, "Evolution and detection of polymorphic and metamorphic malwares: A survey," *arXiv preprint arXiv:1406.7061*, 2014.

[50] E. H. Shortliffe, *MYCIN: a rule-based computer program for advising physicians regarding antimicrobial therapy selection.*, Tech. Rep., Stanford Univ Calif Dept of Computer Science, 1974.

[51] B. Subba, S. Biswas, and S. Karmakar, "Intrusion detection systems using linear discriminant analysis and logistic regression," *2015 Annual IEEE India Conference (INDICON)*. IEEE, 2015, pp. 1–6.

[52] M. Szczepański, M. Choraś, M. Pawlicki, and R. Kozik, "Achieving explainability of intrusion detection system by hybrid oracle-explainer approach," *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[53] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," 2009, pp. 1–6.

[54] A. Ultsch and H. P. Siemon, "Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis," *Proceedings of the International Neural Network Conference (INNC-90), Paris, France, July 9–13, 1990*. 1990, vol. 1, pp. 305–308, Kluwer Academic Press.

[55] M. Vasighi and H. Amini, "A directed batch growing approach to enhance the topology preservation of self-organizing map," *Applied Soft Computing*, vol. 55, 6 2017, pp. 424–435.

[56] C. S. Wickramasinghe, K. Amarasinghe, D. L. Marino, C. Rieger, and M. Manic, "Explainable Unsupervised Machine Learning for Cyber-Physical Systems," *IEEE Access*, vol. 9, 2021, pp. 131824–131843.

[57] C. Wu, A. Qian, X. Dong, and Y. Zhang, "Feature-oriented Design of Visual Analytics System for Interpretable Deep Learning based Intrusion Detection," *2020 International Symposium on Theoretical Aspects of Software Engineering (TASE)*. IEEE, 2020, pp. 73–80.

[58] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Applied soft computing*, vol. 10, no. 1, 2010, pp. 1–35.

[59] S. Zanero, "Analyzing TCP traffic patterns using self organizing maps," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3617 LNCS, 2005, pp. 83–90.

[60] S. Zanero and G. Serazzi, "Unsupervised learning algorithms for intrusion detection," *NOMS 2008 - IEEE/IFIP Network Operations and Management Symposium: Pervasive Management for Ubiquitous Networks and Services*, 2008, pp. 1043–1048.

[61] G. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, 2000, pp. 451–462.