Enhancing Self-Organizing Maps with Numerical Criteria: A Case Study in SCADA Networks

by

Tianming Wei
B.Sc., Nankai University, 2011

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Enhancing Self-Organizing Maps with Numerical Criteria: A Case Study in SCADA Networks

by

Tianming Wei
B.Sc., Nankai University, 2011

Supervisory Committee

Dr. Sudhakar Ganti, Co-Supervisor
(Department of Computer Science)

Dr. Yvonne Coady, Co-Supervisor
(Department of Computer Science)

Dr. Stephen Neville, Outside Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Sudhakar Ganti, Co-Supervisor
(Department of Computer Science)

Dr. Yvonne Coady, Co-Supervisor
(Department of Computer Science)

Dr. Stephen Neville, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Self-Organizing Maps (SOM) can provide a visualization for multi-dimensional data with two dimensional mappings. By applying unsupervised learning techniques to SOM representations, we can further enhance visual inspection for change detection. In order to obtain a more accurate measurement for the changes of self-organizing maps beyond simple visual inspection, we introduce the Gaussian Mixture Model (GMM) and Kullback-Leibler Divergence (KLD) on top of SOM trained maps. The main contribution in this dissertation focuses on adding numerical methods to SOM algorithms, with anomaly detection as example domain. Through extensive traced-based simulations, it is observed that our techniques can uncover anomalies with an accuracy of 100% at an anomaly mixture-rate as low as 12% from the CTU-13 dataset. Tuning of the KLD threshold further reduces the mixture-rate to 7%, significantly augmenting visual inspection to assist in detecting low-rate anomalies.

Suitable hierarchical and distributed SOM-based approaches are also explored, along with other approaches in the literature. Hierarchies in SOM can show the correlations among the neural cells on the self-organizing maps. In order to obtain a higher accuracy for anomaly detection, a new dimension of labels is suggested to be added in the second layer of SOM training. Also for more general distributed SOM-based algorithms, we investigate the use of principal component analysis (PCA) for

the separation of dimensions. With the transformed dataset from PCA, the inner dependencies can be reserved in a manageable scale.

As a case study, this dissertation uses a SOM-based approach for anomaly detection in Supervisory Control And Data Acquisition (SCADA) networks. We further investigate the use of SOM for the Quality of Service (QoS) in the scenario of wireless SCADA networks. Solving the problem of long computing time of optimizing the cached contents, the new SOM-based approach can also learn and predict the sub-optimal locations for the caching while maintaining a prediction error of 28%.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Sudhakar Ganti, Dr. Yvonne Coady, for mentoring, support, encouragement, patience, motivation, and immense knowledge.

Dr. Ulrike Stege, Dr. Steve Evans, Dr. David Capson, for their generosity, kindness and humanity which enlightens me from the dark.

Mitacs, for funding me with an internship.

My families, all the schoolmates and friends, for supporting me in the low moments.

If you believe in yourself and have dedication and pride and never quit, you will be a winner. The price of victory is high but so are the rewards.
Paul Bryant

DEDICATION

To the people who shared their life with me.

Chapter 1

Introduction

A Self-Organizing Map (SOM) is a type of Artificial Neural Network (ANN) that is used in many applications including visualization, data representation, data reduction, density reduction and classification. A SOM is an unsupervised learning algorithm that keeps the topological mapping between input and output, which is useful when dealing with a large set of data with varying dependencies. The SOM algorithm can be used as part of a system to classify and detect abnormal data rate traffic flows, which can make the system capable of finding unknown attacks and provide amazingly beautiful visualizations at the same time. However, SOM also has the scalability problem for larger networks with more than thousands of network flows which makes it difficult for human visual inspections. This dissertation focuses on enhancing change detection in SOM algorithms through the addition of numerical methods. As an example, we consider anomaly detection in Supervisory Control And Data Acquisition (SCADA) Networks. In this Chapter, we will introduce the basics for SOM and the application domain for our case study, the SCADA Networks. The basic SOM algorithm will be evaluated in Chapter 2 and the numerical methods will be discussed in Chapter 3.

1.1 Basics for Self-Organizing Maps

Building a SOM consists of two phases: *learning* and *classification*. In the *learning phase*, the SOM is trained within certain epochs by collecting known data representing normal behavior and abnormal behavior of the network to produce a traffic model consisting of clusters such that each packet passing through the network can be placed

in one of these clusters. An input vector file is created during each data collection window and is then processed through the SOM to classify each vector into one of the clusters as normal or abnormal. The *classification phase* of the data can take place in subsequent epochs. Visualization of the cluster on a 2-dimensional map, which could make the anomaly detection more effective and intuitive, can be built on top of the existing methodologies to aid the human eye. The main procedure of a SOM algorithm includes [7]:

1. *Initialize weights:* Suppose the number of variables in the input dataset is n and size of the training neural network is p by q, then random values will be assigned to the network weights $w_{ij}(k)$, where $i = 1, 2, ..., p.$, $j = 1, 2, ..., q.$ and $k = 1, 2, ..., n.$. In most cases random initialization is sufficient as the SOM will inevitably converge to a final mapping.

2. *Looping through T training epochs:* (a) Select a sample x from the input data set. (b) Find the "winning" neuron for the sample input by calculating the Euclidean Distance d and find the neuron with the minimum distance.

$$d = \min_{ij}\{\|x - w_{ij}\|\}, \tag{1.1}$$

   where $\|\cdot\|$ is the Euclidean norm and w_{ij} is the network weight.

   (c) Adjust the weights of nearby neurons with:

$$w_{ij}^{t+1} = w_{ij}^{t} + \eta^t K^t(i,j)\{x^t - w_{ij}^{t}\}, \tag{1.2}$$

   where η^t is the learning rate at epoch t, and $K^t(i,j)$ is a suitable neighborhood function [8].

3. *Grouping:* Groups of similar neurons in the final map are generated in this process.

Figure 1.1 shows an example of training with a SOM algorithm on a sample data with three features (RGB values), which can be colored based on the RGB values. In this figure, first the map is randomly initialized with various RGB values. After looping through 20 epochs, a new map can be obtained in which the neural cells with similar colors become closer to each other on the map. However, the trained map cannot guarantee the position for a specific color unless a fixed seed number is chosen

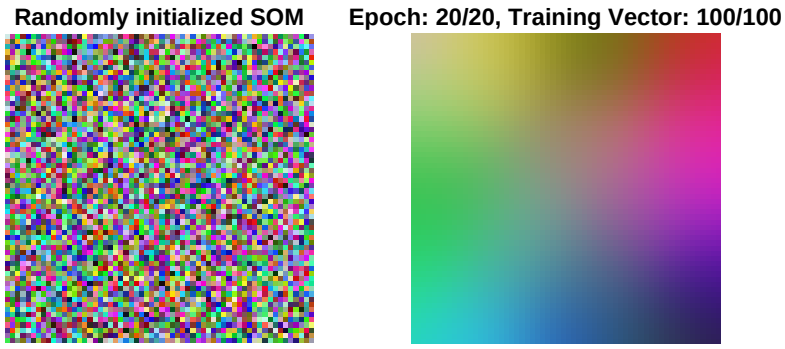


Figure 1.1: A Simple Example of SOM

for random initialization in the SOM training phase. This is important and can be utilized in order to pinpoint the anomaly in the network traffic on the trained SOM map. As we are using a SCADA network as a case study, the sections below will introduce SCADA networks in general.

1.2 A Case Study Scenario: SCADA Networks

Remote monitoring and control of critical infrastructure (e.g., power generation, oil refineries, manufacturing plants etc.) has been possible due to the advent of networking technologies in the last few decades. Industrial Control Systems (ICS) is a major segment within these operational technology sectors. Most ICSs are managed via Supervisory Control And Data Acquisition (SCADA) networks that are commonly deployed to aid the operation of such large industrial infrastructures, including some considered essential for our society, such as water treatment and power generation facilities. These systems are used by operators in modern industrial facilities to continuously monitor and control plant operations.

SCADA systems enable network-based monitoring, communication and/or control of processes (or plants) in various industrial sectors including electrical power distribution, energy, oil and gas, waste water and water. These systems serve as the backbone of much of Canada's critical infrastructure. Security compromises of

SCADA systems allows malicious attackers to gain control of the industrial processes in question - with devastating results. There are numerous real world examples of SCADA network compromises. In Australia, a disgruntled former employee gained remote unauthorized access to a waste water plant and discharged 800,000 liters of sewage into the water system. In another example, the well-known Stuxnet worm highlighted the potential of gaining remote control of nuclear plants. Industrial control infrastructure and systems in Canada and overseas are increasingly the target of cyber security attacks. Historically, SCADA networks were based on the closed Public Switched Network (PSN) but today many are based on open Local Area Network (LAN)/Wide Area Network (WAN) technologies and in many cases, unintentional internet connections can arise when SCADA systems are connected to a corporate networks to enable reporting.

A typical SCADA network architecture will consist of lower-level devices such as Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs) which control physical processes, as well as HMI (Human Machine Interface) software which controls and monitors the PLCs. The HMI, associated databases, data acquisition server and historians are software systems often installed on Windows platforms and connected to an IP-based network. SCADA architectures use standard protocols to communicate with PLCs including protocols such as Modbus, Ethernet/IP and DNP3. These protocols have been refined so that they operate over typical TCP/IP based networks.

SCADA systems have evolved over time in terms of the capabilities of their sensors and actuators as well as in their network topologies. SCADA network topologies have moved from simple point-to-point links to arbitrary mesh-type network topologies that include fixed and wireless links to support large numbers of nodes and overlapping networks. Given their critical nature, security plays a very important role, as the impact of attacks could be catastrophic [9] in these infrastructures.

Figure 1.2 shows a general architecture for a SCADA network, including 4 sub-networks (Field-Level Network, Control Network, Plant Network and Corporate Network). As the "Corporate Network" is connected with the outside Internet, the requests from this sub-network must be treated with care. Also, a large quantity of network traffic in SCADA networks will have real-time requirements and thus, significant attention should be given to the methods to detect the anomalies as well as locations to deploy such tools. Differences in algorithms and deployment methods may significantly impact the accuracy and efficiency of threat detection.

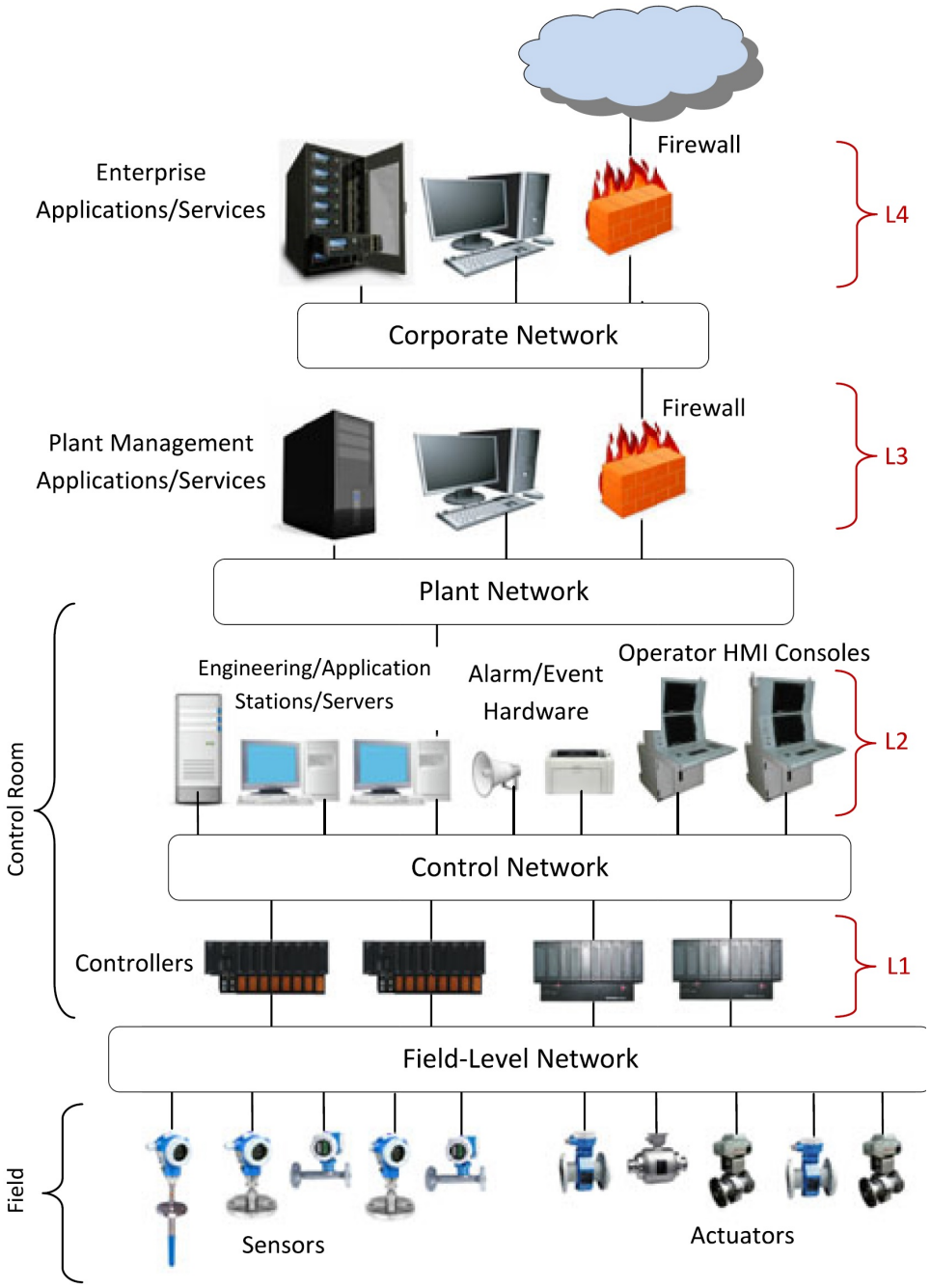


Figure 1.2: Scada Scenario [1]

The evolution of SCADA architectures towards facilitating remote operator control over a network has improved the efficiency and lowered the costs of organizations responsible for the operation and maintenance of such networks. However, connecting

SCADA networks to other IP networks has made the networks and their systems susceptible to security threats as the original SCADA protocols were not designed to be secure protocols. The assumption was that they would run in a closed and restricted network environments. It can be difficult to sufficiently vet upgraded devices so as to sufficiently minimize the risk associated with operationally deploying them.

The above necessitates early and rapid detection of cyber threats to SCADA networks. The majority of existing firewalls, Intrusion Detection Systems (IDS) and cyber security/forensic tools, however, do not have strong support for detection of cyber threats to SCADA infrastructure due to limited support for SCADA protocols and architecture. Studies also indicate that there is increased attention by the research, academic and vendor community to develop innovative techniques for detection of cyber threats against SCADA networks [10]. In one such effort [11], a model based detection system was developed that re-constructs the protocol behavior in SCADA networks using real network traffic.

In general, cyber threat detection can be divided into two categories: signature-based detection and anomaly-based detection (also referred to as behavioral based detection). The majority of security monitoring systems (including IDS, Anti-virus systems etc) implement a signature-based approach to threat detection. This approach maintains a database of known threat signatures (e.g. byte value patterns in a series of packets). They detect threats by examining incoming packets against database signatures to find a match. Signature-based systems are effective in detecting attacks for known cyber threats. However, signature-based systems are rendered ineffective in cases where the data stream is encrypted (e.g, command control channel of a Botnet) and where new zero-day threats are generated for which signatures have not yet been determined.

Anomaly detection techniques have been used widely in different fields including the financial sector and medical research [12]. The majority of techniques generally establish baseline behavior for the data being examined and then proceed to detect deviations from the norm [13]. When applied to cyber-threat detection, anomaly detection schemes generally operate on the principle that network traffic behavior of devices in a network can be baselined to understand their normal traffic generation patterns. They then look for "abnormal" traffic patterns [14, 15].

Although the importance of SCADA systems has been recognized for some time [16], efforts to investigate network security related issues in SCADA environments have been relatively limited [17, 18]. Igure, et al. [18] identify several security challenges

that have to be addressed for SCADA networks such as: access control, firewalls and intrusion detection systems, protocol vulnerability assessment, cryptography and key management, device and operating system security, and security management.

This research will examine all the above questions in the context of SCADA networks and propose solutions to effectively detect cyber threats. We focus on the anomaly detection techniques that aim to develop effective and efficient methods to detect attacks such as denial of service, scans, worm and so on based on traffic profiling.

1.3 Research Questions

Specifically, the research questions addressed in this dissertation evaluated enhancements to an off-the-shelf SOM-based approach using SCADA networks as a case study. The questions addressed are:

1. Can a SOM-based approach provide sufficient visualizations in a case study of SCADA networks for anomaly detection purpose (Chapter 2)?

2. Can further numerical criteria for change detection enhance this SOM-based approach (Chapter 3)?

3. Can hierarchical and distributed SOMs provide further insights into traffic control and analysis with large amounts of data (Chapter 4)?

4. Can SOMs enhance self-adapting caching strategies by learning about the history of optimal caching strategies in the extended case study of wireless SCADA networks with mobile clients (Chapter 5)?

The case studies, methodology, results and analysis for Research Question 1 (Chapter 2) provide the foundation upon which the shorter case studies for Questions 2 and 3 build (Chapters 3 and 4).

1.4 Dissertation Outline

The remainder of this dissertation is organized as follows.

Chapter 2 gives the self-organizing map (SOM) based approach. In our case study of detecting the traffic anomalies in SCADA networks, SOM is evaluated through extensive real-trace based simulations and shown as a useful method.

Chapter 3 further extends the SOM-based approach with numerical methods by applying the Gaussian Mixture Model (GMM) and Kullback-Leibler Divergence (KLD) and provides a numerical criteria for comparing different SOMs.

Chapter 4 discusses the hierarchical design of SOM training. With new labels being added to the second layer of SOM training, the anomalous and normal inputs can be clearly separated. Moreover, pointing out the loss of dependency with feature separated SOM training. A new principal component based SOM training is further developed and justified to conserve the inner correlations in the original dataset.

Chapter 5 investigates the use of SOM in the extended case study of wireless SCADA networks. The SOM algorithm can be used as an efficient approximating method and improve the performance for content caching in wireless SCADA networks.

Chapter 6 concludes the dissertation and enumerates avenues of future work for further development of the concept and the SOM-based applications.

Chapter 2

Self-Organizing Maps Case Study in SCADA Networks

As a case study, this dissertation uses a SOM-based approach for anomaly detection in Supervisory Control And Data Acquisition (SCADA) networks. More details on the background of SCADA security and anomaly detection are first introduced in this Chapter and then gives the self-organizing map (SOM) based approach. In this case study, SOM is evaluated through extensive real-trace based simulations.

2.1 Security Background for SCADA Networks

SCADA systems control many infrastructures such as power grids, gas pipes, water circulation and other natural resources, which all play a vital role in daily life. However, vulnerabilities in the hosts of SCADA systems have increased recently due to their increasing inter-connectedness with IP networks and the World Wide Web. In most cases, intrusions into SCADA systems will cause a change, or an *anomaly* in the communication among different components of the system.

Intrusion detection is defined as the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible violations, or imminent threats of violation, of security or acceptable use policies [19]. Two major approaches of intrusion detection are *signature based* and *anomaly based*. Signature detection matches traffic to a known misuse pattern while the anomaly detection works on the abnormalities in the observed monitoring data. There are other methods which fall between the two approaches, for example, the probabilistic

and specification based approaches [20]. These approaches embed probabilistic modeling or model allowable system traffic patterns, respectively. While misuse based detection methods have reached somewhat of a saturation point, the primary focus of many current research initiatives has been in writing signatures or enhancements of signature matching using state machines. Network traffic analysis has been shown to be an effective tool to determine regular versus anomalous behavior. This approach can be leveraged to detect intrusions in the network.

The field of traffic measurement is well established in IP networks. For example, Cisco's Netflow protocol is one popular way to collect detailed IP traffic information. However, the highly decentralized nature of the Internet along with information hiding properties in various layers can make it difficult to determine what is normal versus what is abnormal in terms of traffic patterns. Other ways of collecting traffic information include traffic matrix estimation, traffic dynamics measurements, traffic mix measurements and active versus passive monitoring.

In the case of a SCADA network, which is a controlled environment, traffic features are different from traditional IP networks. Some of the traffic flows in a SCADA network could be static and periodic, allowing us to accurately apply general probability models for the traffic flows in the SCADA networks. As established in [9], SCADA traffic exhibits traffic periodicity due to regular polling and updating of data by the Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTCs). Since periodicity and size of the traffic can be defined, this separates SCADA traffic from other communications in the whole network. Though some SCADA communications are non-periodic, many intrusion attempts can disturb the periodicity of the SCADA traffic, and therefore be detected. Packet traces were used to create flow information and spectral techniques (e.g., Fast Fourier Transform (FFT)) to recognize the periodicity of SCADA traffic, with anomalies identified as any deviations of the traffic periodicity.

Other works have also shown promise in detecting less frequent network traffic patterns [21]. By leveraging non-uniform two-stage traffic sampling to identify "elephants from mice", a novel buffering scheme using two-stage filtering scheme was able to identify less frequent traffic patterns and provided a caching method for frequent network patterns. The sampling techniques handle the problem of an enormous data set with improved scalability.

Traditional use of Intrusion Detection Systems (IDS) has also been applied to anomaly detection in SCADA networks. Most of the detecting schemes in IDSs require

a traffic monitoring and a database of traffic traces needs to be maintained. However, in order to detect the anomaly in the network, another database for attack signatures is also needed and updated periodically. For example, the most popular open source IDS project, SNORT [22], has more than 50 rules for the MODBUS protocol [11], which is generally used in SCADA networks, and more custom rules can be added. By using this kind of signature-based intrusion detection, only known attacks can be detected. Therefore we use a Self-Organizing Map (SOM) based approach to detect unknown attacks with acceptable training time from neural networks.

One of the challenges in utilizing anomaly detection techniques for cyber threat detection in regular networks is that the baseline network traffic behavior can be quite noisy in nature, making it difficult to detect anomalies accurately. As a result, some proposed anomaly detection schemes in cyber security suffer from high false positives [23]. We believe that anomaly detection schemes should work well in a SCADA network environment because such networks have more deterministic and less noisy network traffic. In such environment, it should be easier to detect more reliably cyber threats using anomaly detection schemes. Anomaly detection techniques have a great deal of flexibility when applied to network traffic. For example, there is a need to determine which technique to use (PCA vs Clustering and many others) [24, 25]. There is also a need to determine whether to use a supervised learning (uses training data set) or unsupervised learning technique. There is a need to decide whether to apply the techniques against raw IP packets or against summarized flow statistics. There is also a need to determine which traffic features or statistics should be calculated against which the algorithms will be applied.

Our proposed solution is to leverage neural network mappings, specifically Self-Organizing Maps (SOM) [7] to detect unknown attacks with acceptable training time. Neural networks capable of unsupervised learning can provide a powerful supplement to anomaly detection [26]. After learning the characteristics of normal traffic in the network, the trained neural network can identify anomalies without relying on expectations of what anomalies will look like [26, 27]. The neural network can also encompass attack behaviours within its model of what it believes to be "normal". As one popular approach from the neural networks, SOM can be used for unsupervised learning and can provide a two dimensional map for multi-dimensional data, which makes visual inspection a possibility. In this dissertation, we introduce a SOM-based approach for the application of anomaly detection in SCADA networks. By applying the Gaussian Mixture Model (GMM) and Kullback-Leibler Divergence (KLD) on top

of SOM-trained maps, we accurately identify a threshold mixture-rate for detecting anomalies.

SCADA systems are widely implemented for the control of many national infrastructures such as power grid, gas transportation pipes, water circularization and other nature resource utilizations, which play a vital role in our daily life. However, discovered vulnerabilities in SCADA systems have increased recently due to their increasing inter-connectedness with the IP networks and the world wide Internet. In most cases, the intrusions of the SCADA systems will cause a change, or so called anomaly in the communications among different components of the system. Network traffic analysis has been applied in numerous contexts as an effective tool to determine normal versus anomalous behavior, and thus it can be further utilized to determine whether a incursion occurred in the network.

The field of traffic measurements is well established in IP networks. For example, Cisco's Netflow protocol is a way to collect IP traffic information which has become a widely adopted mechanism to monitor traffic in IP networks. However, the highly decentralized nature of the Internet along with information hiding properties in various layers can make it difficult to determine what is normal versus what is abnormal. Moreover, there are multiple ways of collecting traffic information which should be considered carefully, including traffic matrix estimation, traffic dynamics measurements, traffic mix measurements and active versus passive monitoring.

However, due to the use of a SCADA network in a controlled environment, the traffic features of a SCADA network are different from the traditional IP networks. Some of the traffic flows in a SCADA network could be periodic and with small variations, and general probability models can be also applied on the analysis of the traffic flows in the SCADA networks. In this chapter, we will discuss the general traffic patterns of SCADA networks based on a real traffic trace, and later introduce a neural network based approach for the application of anomaly detection in SCADA networks. Through extensive trace-based simulations such as in Sec. 2.3, the SOM-based algorithm has been justified.

2.2 Traffic in SCADA Networks

The most important reason that the SCADA networks are different from regular IP networks is because the SCADA networks have their own industrial processes and network architectures and protocols. A basic architecture was proposed in [28] on how

to use traffic measurements in SCADA networks. This work suggested placements of traffic monitoring tools at the demarcation boundaries of Internet, Corporate, SCADA and process control sub-networks. As discussed in [9], SCADA traffic can exhibit traffic periodicity due to regular polling and updating of data by the Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTCs). This work suggested that periodicity and size of the traffic can be determined which separates the SCADA traffic from other communications in the whole network. There exists some SCADA communications which are non-periodic, however, the main motivation is that many intrusion attempts can disturb the periodicity of the SCADA traffic. The approach in their work used packet traces to create flow information and spectral techniques (e.g., Fast Fourier Transform (FFT)) to recognize the periodicity of SCADA traffic and thus an anomaly can be detected as a deviation of the traffic periodicity.

Also in the work of [21], which relates to detecting less frequent network traffic patterns, most of the well regulated SCADA traffic is very periodic and regular in nature while other traffic patterns mostly are related to network intrusions or other anomalies. This work also introduced non-uniform two-stage traffic sampling to identify *elephants from mice*, which means that using a buffering scheme, the two-stage filtering scheme can identify less frequent traffic patterns and provides a caching method for frequent network patterns. The novel sampling technique handles the problem of enormous data sets and improves scalability.

2.2.1 A Real Traffic Trace of a SCADA Network

We assume that SCADA traffic has a regular periodic behavior with limited variance. To assert this, we focus on the traffic analysis based on real traffic traces, which were captured by the ICS lab at the 4SICS conference [2]. There could be other models to describe the traffic of SCADA networks [29], but due to the general lack of availability of real SCADA traces, we can only make our assumptions/models based on what we have. To verify the usability of our approach, our method is also evaluated based on different traces in Sec. 3.2. The "Geek Lounge" at 4SICS contains an ICS lab with PLCs, RTUs, servers, industrial network equipment (switches, firewalls, etc), which is shown in Fig. 2.1. These devices are available for hands-on "testing" by 4SICS attendees [2].

The dataset is available for public and can be downloaded from the website [2]. The trace data is provided in the "pcap" format that is compatible with the well-

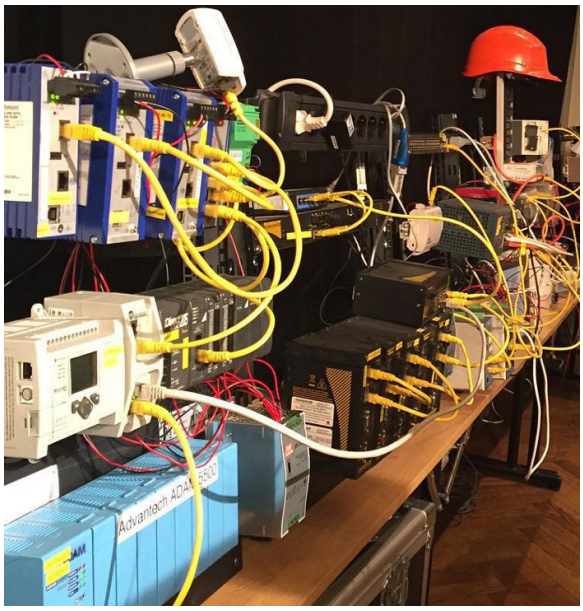


Figure 2.1: Devices in the ICS Lab [2]

known packet analysis software, WireShark [30]. We use WireShark to read the file and do a basic analysis on the topology and traffic pattern of the network.

2.2.2 Topology Graph

Based on the notes provided by the ICS lab [2] which includes the names and usage of specific IP addresses, a topology graph is inferred as shown in Figs. 2.2, 2.3 and 2.4. The numbers on the links represent the number of packets sent on those links as inferred from the pcap files. Most of the communication occur within the sub-network of 10.10.10.x, which is supposed to be the monitored SCADA network. The dominant IP domain in the network is 192.168.x.x, which are switches and routers. Some of the routers can also connect to the outside Internet, for example there is a router which sends DNS requests to the DNS server as shown in both Fig. 2.2 and Fig. 2.3.

The behavior of the attacker/hacker is shown in the context of Fig. 2.4, where an attacker with a new IP address sends packets to multiple IP addresses but with much lower flow-rate compared to that of traffic within the grid. This infers a similar pattern as that of a DoS attack, in which the network will suddenly receive a large amount of connections from different IP addresses. It should be noted that in this real data capture, the attacker was using the TCP protocol with general port number 80 that is different from the Modbus TCP protocol with port number 502 or S7 protocol

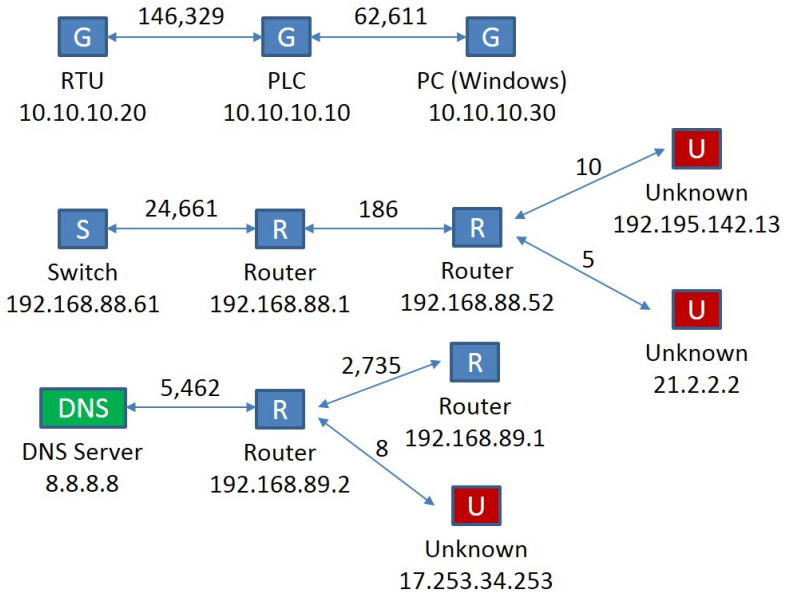


Figure 2.2: Topology Graph without Attackers Part I

with port number 102. It is also noted that the 10.10.x.x network is dedicated to be in the SCADA network and the remaining IPs are in the semi-trusted or corporate zones. As there is no mention of the router connecting these two networks in their experiment document [2], it is assumed that the attacker is not able to send any malicious traffic to the SCADA network.

2.2.3 Statistics on Conversations

Using the software WireShark, a few statistical properties can be extracted easily from the trace data. Some traffic flow conversations are shown in Figs. 2.5. Both TCP and UDP traffic exist in this SCADA network and from these figures it can be inferred that most of the conversations occur between the PLCs and the RTUs that are inside the grid network (10.10.x.x addresses).

From the snapshot of the pcap file shown in Fig. 2.5, multiple traffic statistic can be easily obtained, such as the number of packets, the number of bytes, duration and bandwidth for each traffic flow. These are important features for the network monitoring and traffic flow analysis. In the scope of our work, we focus on the flow-rate analysis for SCADA networks.

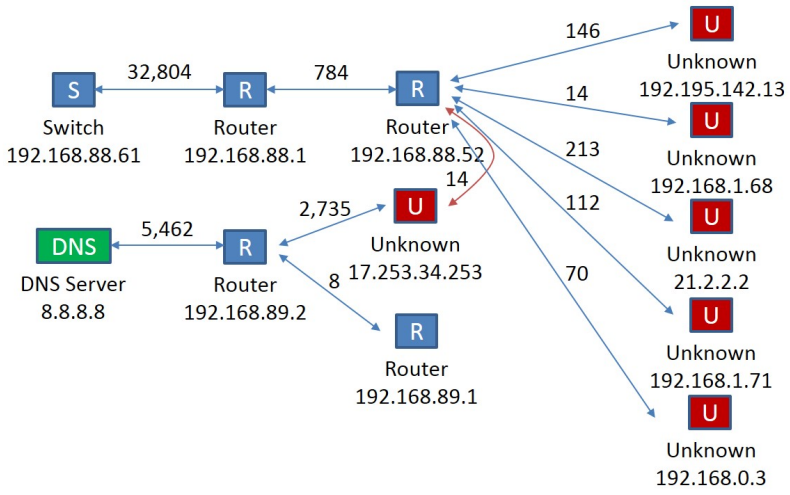


Figure 2.3: Topology Graph without Attackers Part II

2.2.4 Flow-Rate Analysis

Using Wireshark, we also obtain the IO graphs as shown in Figs. 2.6 and 2.7. Figure 2.6 shows the overall network traffic without any attacks. The traffic pattern turns out to be stable with small variations and if we enlarge the scale of the IO graph, we can see a periodic pattern for the network traffic as shown in Fig. 2.7. It means that the devices in this SCADA network will send report packets for every specific period of time.

Figs. 2.8 and 2.9 show the communication rate between the SCADA devices which confirm our previous assumption that the SCADA traffic is more or less regular with a small variance. Fig. 2.8 shows the probability density function of the flow-rate between the 10.10.10.10 and 10.10.10.20, in which a rate at 30 packets per 5 seconds appears to be dominant. However, from Fig. 2.9 it shows a different pattern for the traffic flow between 10.10.10.10 and 10.10.10.30. While the dominant flow-rate is still around 30 to 40 packets per 5 seconds, a normal distribution fitting algorithm is applied in order to see the difference between the real traffic pattern and the one generated from a probability model with limited variance.

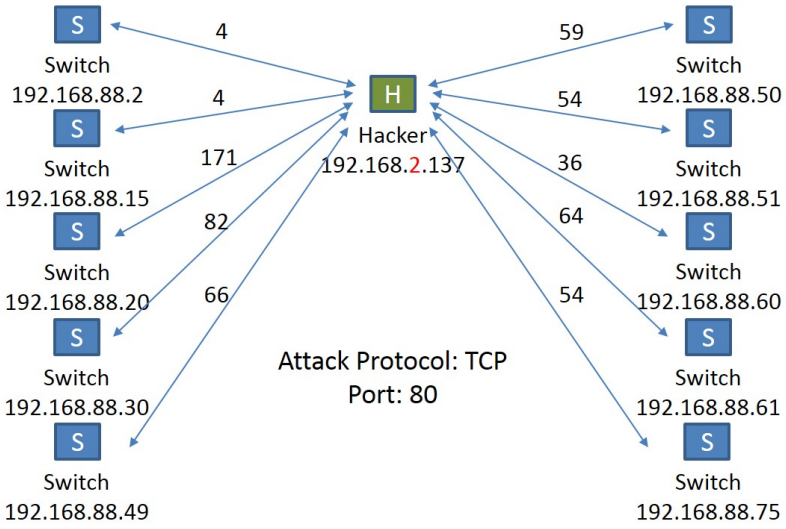
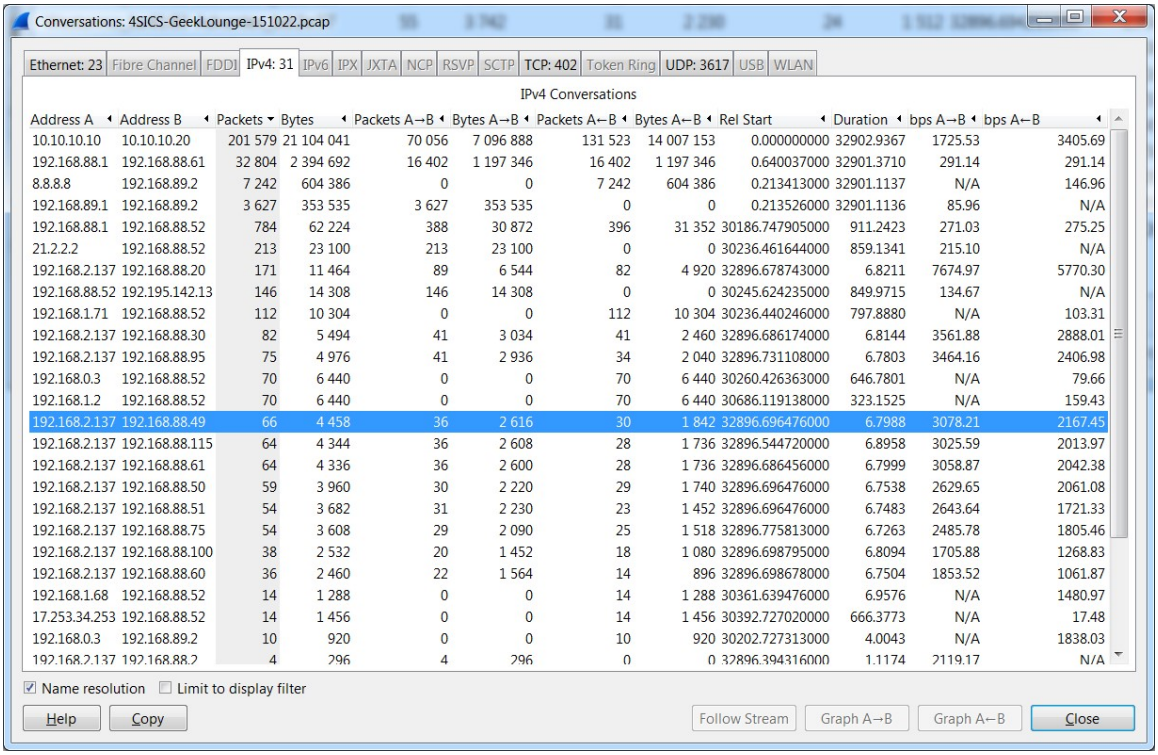


Figure 2.4: Topology Graph with Attackers



Conversations: 4SICS-GeekLounge-151022.pcap

Ethernet: 23 | Fibre Channel | FDDI | IPv4: 31 | IPv6 | IPX | JXTA | NCP | RSVP | SCTP | TCP: 402 | Token Ring | UDP: 3617 | USB | WLAN

IPv4 Conversations

Address A	Address B	Packets	Bytes	Packets A→B	Bytes A→B	Packets A←B	Bytes A←B	Rel Start	Duration	bps A→B	bps A←B
10.10.10.10	10.10.10.20	201 579	21 104 041	70 056	7 096 888	131 523	14 007 153	0.000000000	32902.9367	1725.53	3405.69
192.168.88.1	192.168.88.61	32 804	2 394 692	16 402	1 197 346	16 402	1 197 346	0.640037000	32901.3710	291.14	291.14
8.8.8.8	192.168.89.2	7 242	604 386	0	0	7 242	604 386	0.213413000	32901.1137	N/A	146.96
192.168.89.1	192.168.89.2	3 627	353 535	3 627	353 535	0	0	0.213526000	32901.1136	85.96	N/A
192.168.88.1	192.168.88.52	784	62 224	388	30 872	396	31 352	30186.747905000	911.2423	271.03	275.25
21.2.2.2	192.168.88.52	213	23 100	213	23 100	0	0	30236.461644000	859.1341	215.10	N/A
192.168.2.137	192.168.88.20	171	11 464	89	6 544	82	4 920	32896.678743000	6.8211	7674.97	5770.30
192.168.88.52	192.195.142.13	146	14 308	146	14 308	0	0	30245.624235000	849.9715	134.67	N/A
192.168.1.71	192.168.88.52	112	10 304	0	0	112	10 304	30236.440246000	797.8880	N/A	103.31
192.168.2.137	192.168.88.30	82	5 494	41	3 034	41	2 460	32896.686174000	6.8144	3561.88	2888.01
192.168.2.137	192.168.88.95	75	4 976	41	2 936	34	2 040	32896.731108000	6.7803	3464.16	2406.98
192.168.0.3	192.168.88.52	70	6 440	0	0	70	6 440	30260.426363000	646.7801	N/A	79.66
192.168.1.2	192.168.88.52	70	6 440	0	0	70	6 440	30686.119138000	323.1525	N/A	159.43
192.168.2.137	192.168.88.49	66	4 458	36	2 616	30	1 842	32896.696476000	6.7988	3078.21	2167.45
192.168.2.137	192.168.88.115	64	4 344	36	2 608	28	1 736	32896.544720000	6.8958	3025.59	2013.97
192.168.2.137	192.168.88.61	64	4 336	36	2 600	28	1 736	32896.686456000	6.7999	3058.87	2042.38
192.168.2.137	192.168.88.50	59	3 960	30	2 220	29	1 740	32896.696476000	6.7538	2629.65	2061.08
192.168.2.137	192.168.88.51	54	3 682	31	2 230	23	1 452	32896.696476000	6.7483	2643.64	1721.33
192.168.2.137	192.168.88.75	54	3 608	29	2 090	25	1 518	32896.775813000	6.7263	2485.78	1805.46
192.168.2.137	192.168.88.100	38	2 532	20	1 452	18	1 080	32896.698795000	6.8094	1705.88	1268.83
192.168.2.137	192.168.88.60	36	2 460	22	1 564	14	896	32896.698678000	6.7504	1853.52	1061.87
192.168.1.68	192.168.88.52	14	1 288	0	0	14	1 288	30361.639476000	6.9576	N/A	1480.97
17.253.34.253	192.168.88.52	14	1 456	0	0	14	1 456	30392.727020000	666.3773	N/A	17.48
192.168.0.3	192.168.89.2	10	920	0	0	10	920	30202.727313000	4.0043	N/A	1838.03
192.168.2.137	192.168.88.2	4	296	4	296	0	0	32896.394316000	1.1174	2119.17	N/A

Name resolution | Limit to display filter

Help | Copy | Follow Stream | Graph A→B | Graph A←B | Close

Figure 2.5: Conversation Statistics in File 151022.pcap

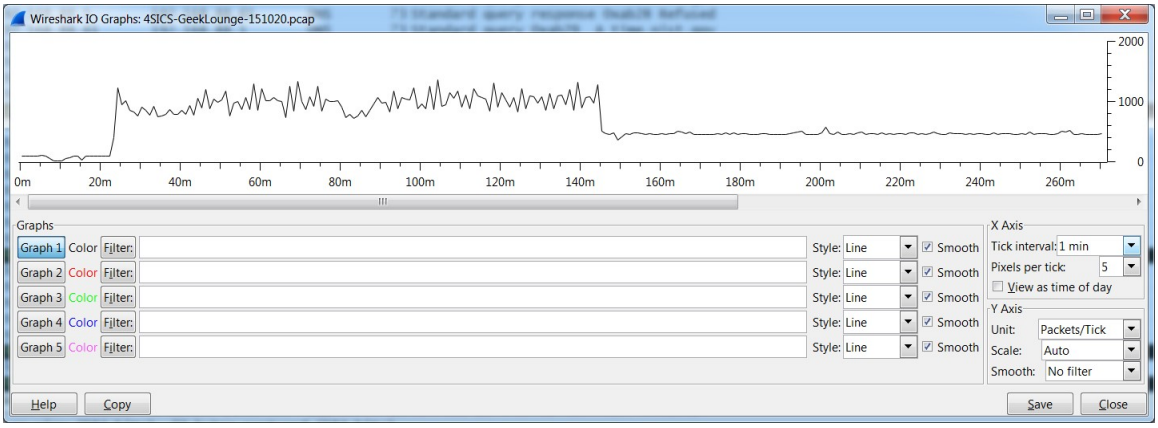


Figure 2.6: Time Sequence IO Graph for File 151020.pcap

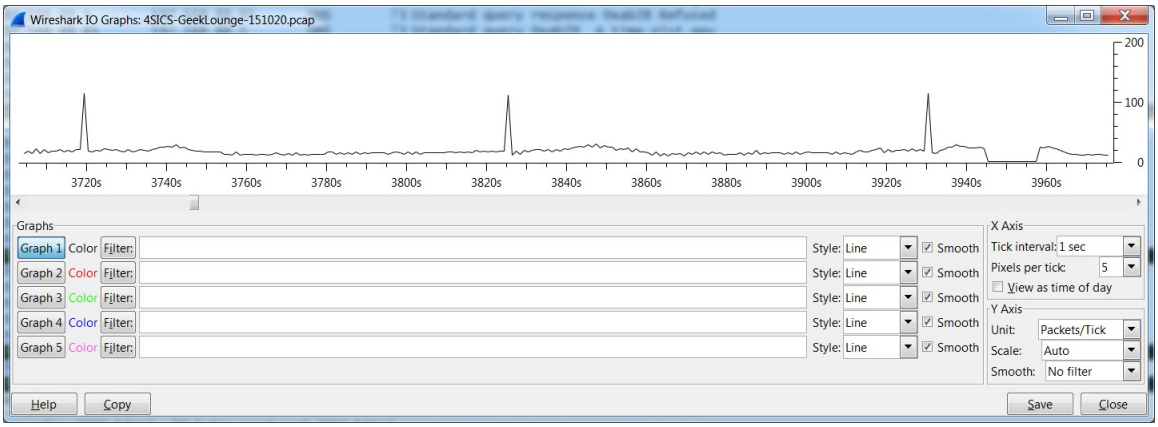


Figure 2.7: Time Sequence IO Graph for File 151020.pcap (Enlarged)

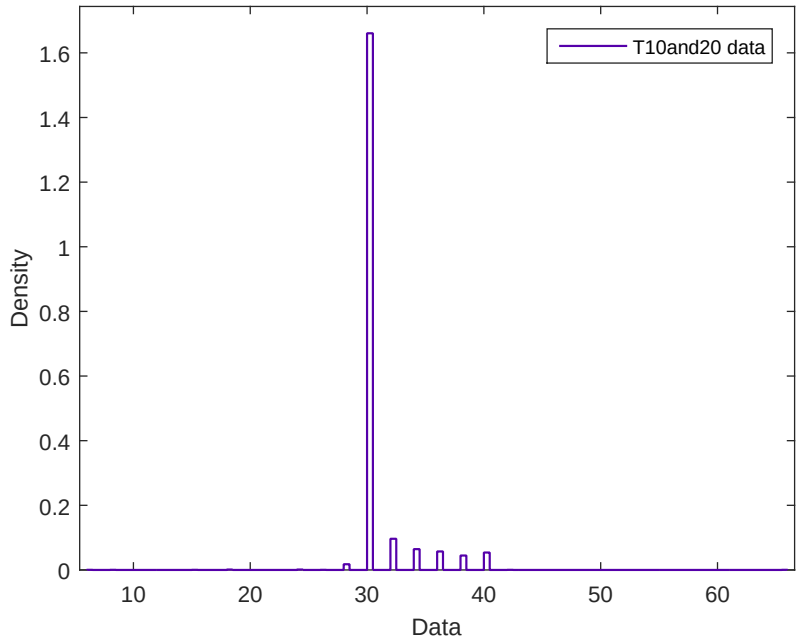


Figure 2.8: Histogram of packets sent for every 5 sec between 10.10.10.10 and 10.10.10.20

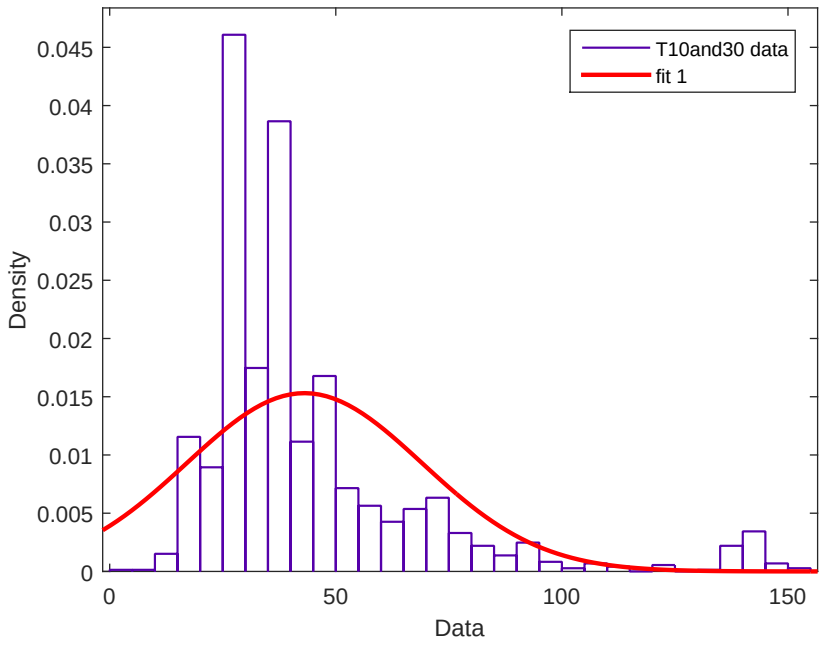


Figure 2.9: Histogram of packets sent for every 5 sec between 10.10.10.10 and 10.10.10.30, Normal Fit

2.3 A Self-Organizing Map (SOM) based Approach

Self-Organizing Maps (SOM) belong to the category of neural-network methods and is popular when it is necessary to map from a multi-dimensional dataset to simple two dimensional representation. As in the monitoring phase of a computer network, especially for the control networks of SCADA systems, there are usually tens of statistical monitoring variables/features existing in the captured log data, it would be challenging to do analysis based on the raw multi-dimensional dataset. However, the SOM algorithm can provide a fast and unsupervised way for data clustering and produce a two dimensional map to visualize the clusters in the dataset.

2.3.1 Simple Clustering Results with SOM Toolbox

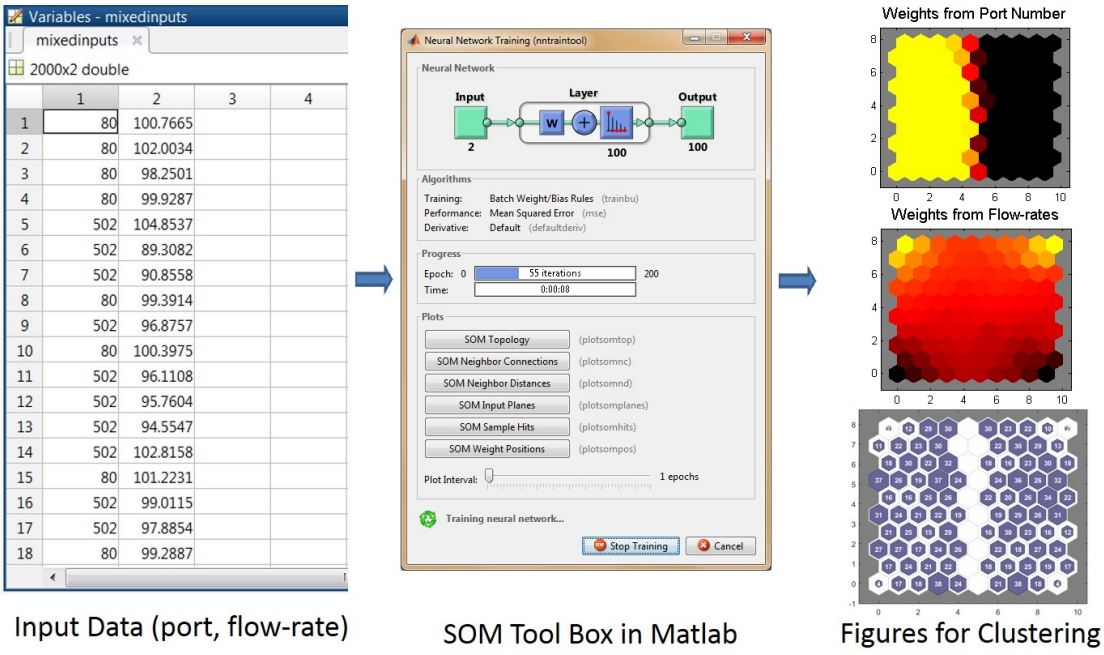


Figure 2.10: Using SOM Toolbox (v2.0) in Matlab (v2015a)

We used the Matlab (v2015a) SOM Toolbox (v2.0) for the training and plotting process which is shown in Fig. 2.10. The raw data needs to be formatted into a file with multiple rows and columns. Each row stands for one sample data and each column stands for each feature captured from the SCADA network. For example, Fig. 2.10 shows a data file with two features, i.e., port number and the number of packets per 5 seconds. This data is processed by the SOM toolbox in Matlab for

training and after a few epochs of training, the SOM map can be plotted and used for further analysis. There are two kinds of output maps, the weights from inputs and the hit map. Each input/feature links to a weight matrix. The weight matrices are updated in each epochs of training and finally there are two colored weights maps generated as shown in Fig. 2.10.

The map on the top shows there are mainly two clusters among the port numbers (502 or 80). However, as there are non-positive values in the weights, we see red and orange colors along the border. The map in the middle shows there is only one cluster among the flow-rates (around 100 packets per second). There will be only one hit map, showing overall clustering results and different clusters appear to be separated on the SOM hit map

2.4 Simulation with SCADASim in OMNeT++

In this section, we will go through a simulated scenario in a controlled environment which can generate a trace dataset for our further evaluation on the SOM based approach for anomaly detection in SCADA networks. Simulating a SCADA system involves modeling the details at various levels of the state of the SCADA entities, the communications between them and the state of the supervised/controlled operation (henceforth called "environment") as shown in Fig. 2.11. Creating a monolithic simulator for all these would not only be costly, but also would produce results in low efficiency, since selecting the right level of detail for each of those depends on our requirements. In fact, some systems may have a part of them not simulated, but emulated or partially implemented using a physical model. Furthermore, when designing a simulator, we should consider how the environment states interact with each other. Some of the existing simulators are compared in Fig. 2.12.

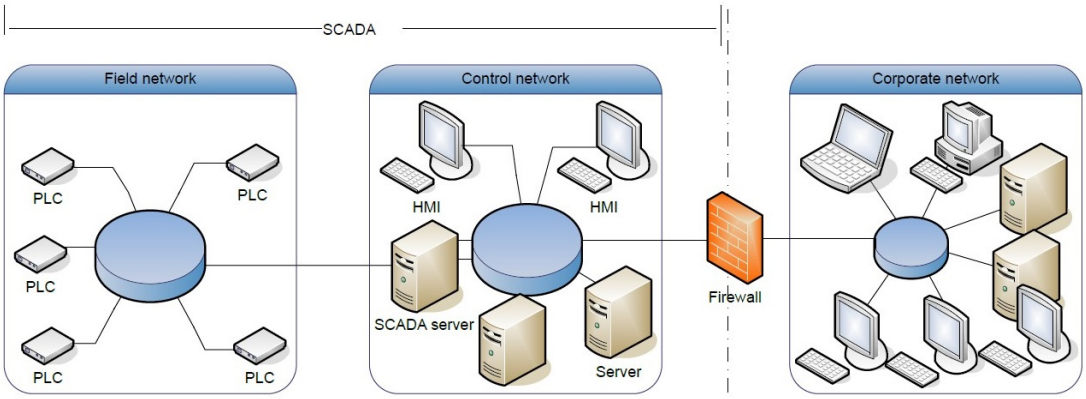


Figure 2.11: SCADA Scenario in Simulation [3]

	Network Simulator	Domain Specific Simulator	Synchronization	Domain
VPNET	OPNET	VTB (Virtual Test-Bed)	Timed stepped (co-simulation coordinator)	Communication Networks of Power Systems
SCADASim	OMNeT++	Built-in modules	SSScheduler module	Malicious attacks in SCADA systems
Wang Chunlei et al.	NS2 (NSE)	CitectSCADA 6.1 (as OPC server) & Real PLC/RTUs	Not mentioned	Security analysis of SCADA systems
EPOCHS	NS2	PSLF, PSCAD/EMTDC	Timed stepped	Electric Power Grid
Lin et al.	NS2	PSLF	Timed stepped (global scheduler)	Smart Grid Applications
DAVIC	OMNeT++	Built-in models	Time-stepped	Energy management algorithms
Chabukswar et al.	OMNeT++	Simulink	Timed stepped (NetworkSim Sceduler)	Attacks on SCADA systems
Neema et al.	OMNeT++	X4 Simulink models	Timed stepped (NetworkSim Sceduler)	Computational experiments in Command & Control
Naturo et al.	NS2	ADEVS	DEVS	Electric Grid

Figure 2.12: Existing Simulators [4]

We are using the SCADASim as the primary simulator. The reason is that SCADASim is totally open source in the language of "c" and it can be easily modified and expanded with new modules on the platform of OMNeT++ [31]. Comparisons on the efficiency of the SOM based algorithms across different simulators are not included in this work. It is a good direction to work on in the future work.

2.4.1 SCADASim

SCADASim [32] is a framework for building SCADA simulations. It provides a modular SCADA modeling tool that also allows real-time communication with external devices using SCADA protocols. This novel framework provides:

1. A modular, extensible, and flexible tool to model SCADA simulations. The SCADASim simulator provides a set of modules that represent SCADA components such as RTU, PLC, MTU, and protocols (such as Modbus/TCP and DNP3). Such modules and protocols can be easily extended, compounded into other modules and used in any SCADASim simulation.

2. The integration of external components into the simulation simultaneously. The SCADASim introduces the concept of gates. A gate is an object that links the external environment with the simulation environment.

3. The possibility of testing attack scenarios seamlessly. SCADASim supports four main types of attacks:denial of service, man-in-the-middle, eavesdropping, and spoofing. Users can easily create attacks to run inside the simulated environment.

SCADASim is built on top of OMNET++ [31], a discrete event simulation engine. OMNET++ consists of modules that communicate with each other through message passing. Various traffic profiles are provided to create communication scenarios of a SCADA network. Network attack scenarios can also be created using the flexible framework. This framework is used to simulate SCADA networks and the SCADASim can also export packet traces in "pcap" format.

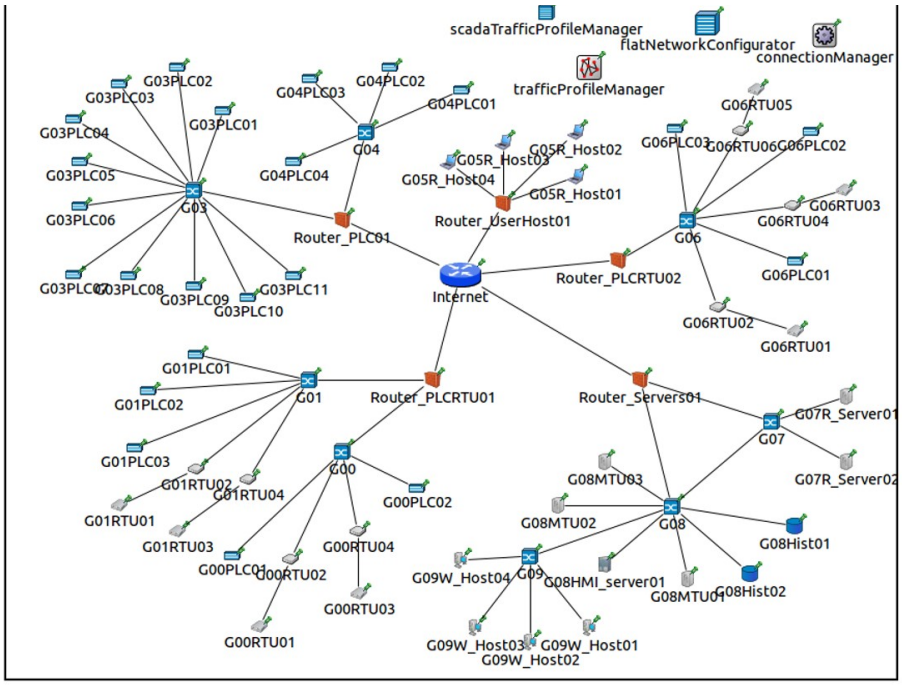


Figure 2.13: Scada Scenario in Simulation

2.4.2 Flow-Rate Detection Method in OMNeT++

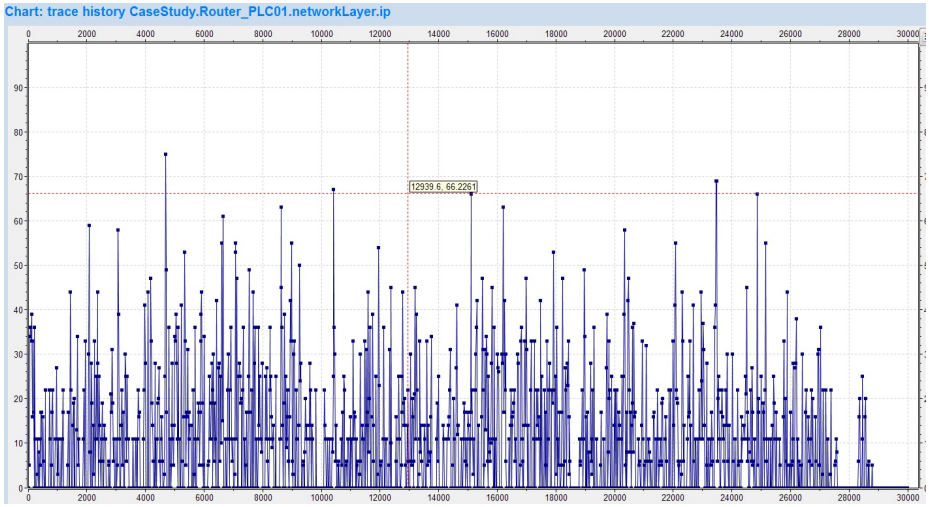
As indicated, SCADASim [32] framework in OMNET++ [31] is a comprehensive suite that is used to simulate SCADA networks and test the proposed SOM based anomaly detection method. As described in previous sections, some of the earlier work concludes that rate-based anomaly detection is a viable solution [9]. Hence we first tested that the SCADASim works correctly in the Ubuntu environment and then incorporated a data rate monitoring method in the routers to monitor and record the traffic flow rates. A sample topology of the network is shown in Fig. 2.13, in which the firewall routers are implemented to be able to record statistics for the number of packets passing through. This simulation scenario is also modified to mark the PLCs, MTUs, HMIs and Workstations appropriately (see Fig. 2.15).

As shown in Fig. 2.14, initial results are obtained by varying the record time interval δt. We used averaging techniques (moving average) to smooth the observed data rates. As shown in Fig. 2.14, the results become more stable when the δt becomes larger. Based on the flow rate observed (i.e., the number of packets for each flow within a certain time δt), thresholds (e.g., the average and maximum flow rates from history data) can be set for the anomaly detection. If the flow rates suddenly exceed the threshold, then there might be an attack in the SCADA network. Various rate monitoring and recording (e.g., non-uniform sampling) methods can be implemented for scalability in future and tested out very easily by changing the algorithms.

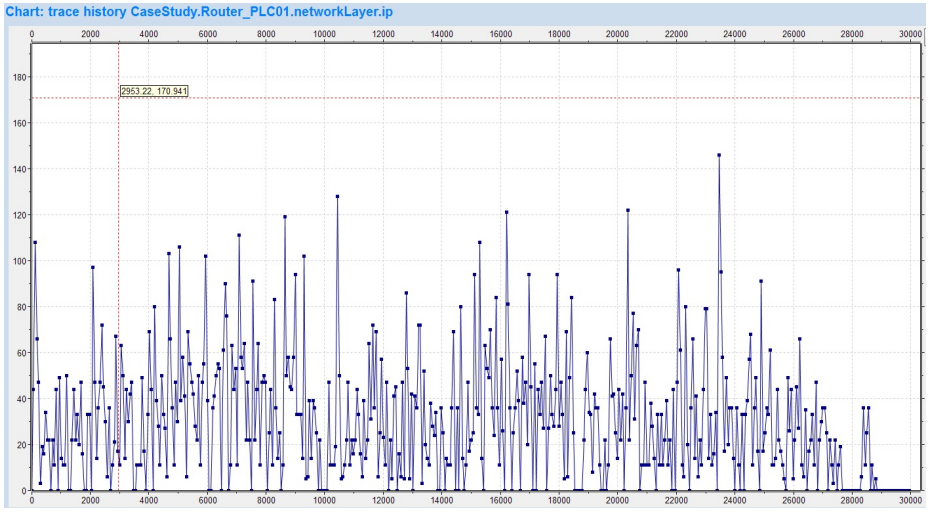
2.4.3 Scenario with Attack

Built on top of SCADASim, we further created a simulation scenario in which a DoSZombie is manually set as shown in Fig. 2.15. We assume that the communication between the hosts on the corporate side and the ones on the field side are using ModBus TCP protocol on port number 502 and also implement pairs of general web clients/servers which communicate using the TCP protocol on port number 80. In order to see the traffic pattern from the simulation, an IO graph (number of packets every 5 seconds v.s. time) is obtained that is shown in Fig. 2.16. The sources are setup such that they start randomly with a uniform distribution of (0,60) seconds and an attacker starts at about 200 seconds. The traffic without attacks has small variations after the initial starting phase.

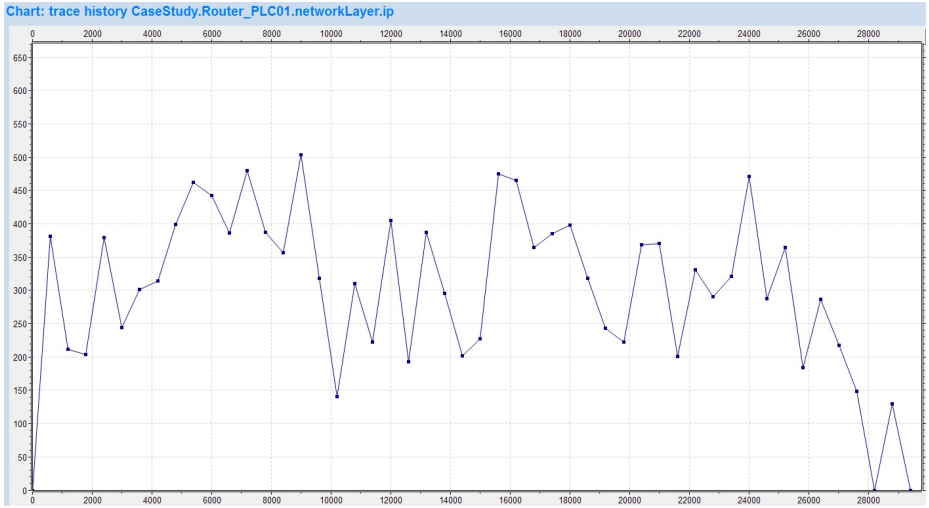
With the incorporation of traffic monitoring probes into the simulator, we are now ready to simulate various network scenarios using the SCADASim. The simulation data will be further analyzed using the SOM technique to verify the working of SOM based anomaly detection.



(a) Record Time Interval 20 s



(b) Record Time Interval 60 s



(c) Record Time Interval 600 s

Figure 2.14: Data Rates observed at various intervals

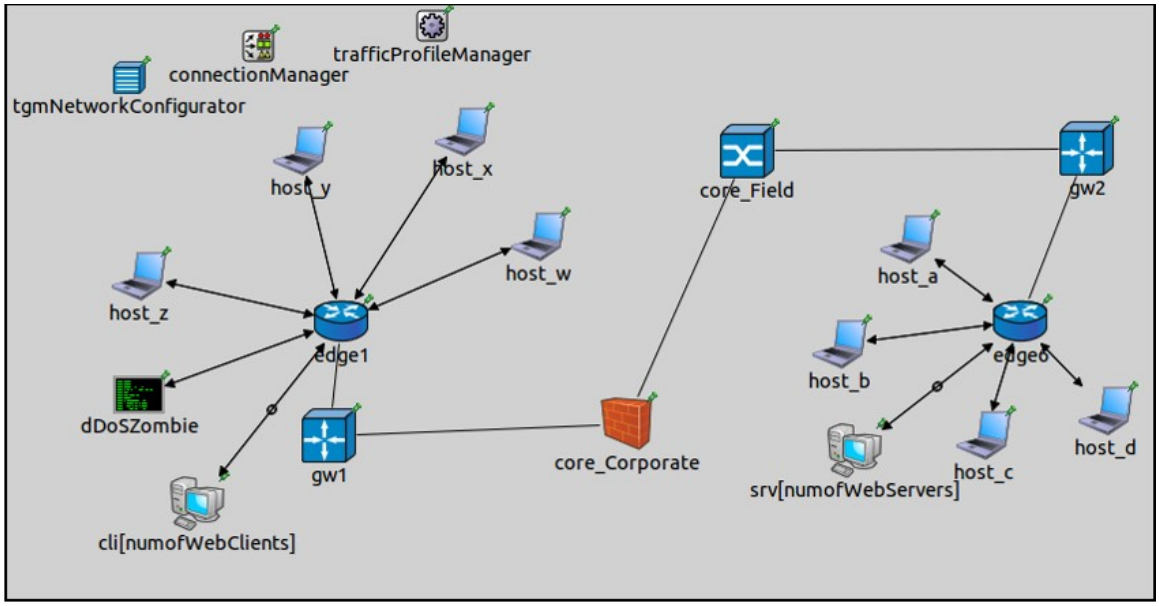


Figure 2.15: Simple Scenario in Simulation

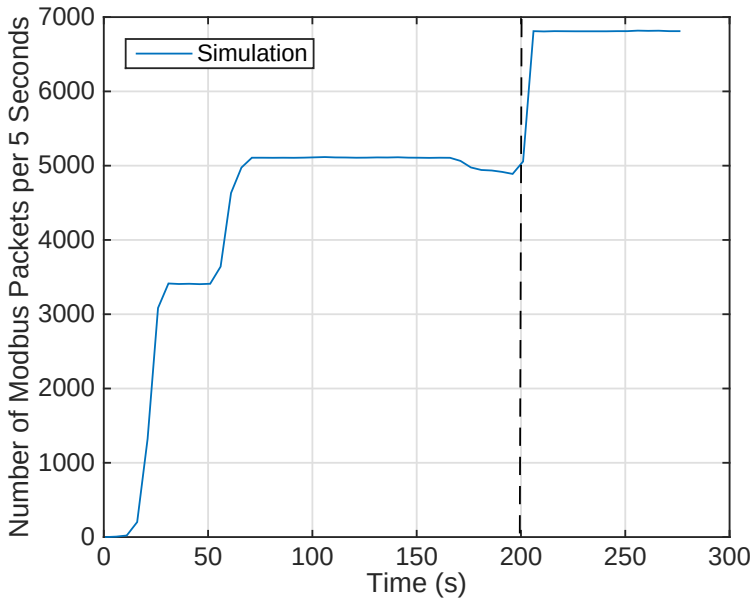


Figure 2.16: Time Sequence IO Graph from OMNeT++ Simulation

2.5 Evaluation with SOM-based Algorithm

In this section we evaluate the proposed SOM-based anomaly detection method. Self-Organizing Maps have been used as an unsupervised learning algorithm. Unsupervised learning is the task of inferring any hidden structures in unlabeled data. A Self-Organizing Map consists of components called nodes or neurons. Associated with each node are a weight vector of the same dimension as the input data vectors, and a position in the map space. The usual arrangement of nodes is a two-dimensional regular spacing in a hexagonal or rectangular grid. SOM is a topographic organization of the data in which nearby locations in the map represent inputs with similar properties. Thus SOM is able to project high-dimensional data to a lower dimension, typically 2D. In order to organize the data, the SOM involves an iterative learning process by which the output gets self-organized and feature map between input and output is realized. The adaptive learning occurs during various epochs of the self-organizing phase. For example, assuming that the initial dataset presented to the SOM represents the normal traffic in a network, then the SOM gets trained say in X epochs, which is normally within 200 epochs and can be also set manually. After the training phase, if any anomalous traffic (attack) pattern is presented to the SOM, it would be able to detect this by a change in the topological map. Our results clearly indicate that the SOM algorithm can detect the anomaly and pin-point the change in the traffic of SCADA network. Also, in order to apply a numerical analysis on the different variations of self-organizing maps, a GMM and KLD based algorithm is applied. Our approach is verified through extensive experiments in Sec. 3.1.3.

2.5.1 Traffic Features

For SOM based anomaly detection, we start with four network traffic features: 1) Source IP addresses; 2) Destination IP addresses; 3) Port numbers; 4) Traffic flow-rates, the number of packets per 5 seconds.

With sufficient dataset from simulation or real traces, one could also employ more features such as the number of bytes per minute, the number of simultaneous TCP connections, the number of unique IP addresses and so on as inputs for the SOM algorithm. However, it should be noted that more features could result in a longer training time. Thus we focus on the four features listed above. In the later Sec. 3.2, our approach is applied and tested based on another traffic trace for a general IP

network. In the new traffic trace, more features/inputs are provided and can be utilized for SOM training.

2.5.2 Weight and Hits Maps

The SOM algorithm keeps the weight matrix natively for each feature in the training phase. The weight matrix is of the same size as that of the map. SOM combines the weights belonging to multiple features in order to determine the clustering of the nodes. If we draw a coloured figure based on the value in the weight matrix, we can roughly see the clusters on each weight matrix. For example, Fig. 2.17 shows the training results with the data set that contains the communication between 3 source IP addresses and 2 destination IP addresses and it is obvious to see 3 distinct clusters in the weight matrix for the source IP addresses (Input 1) and 2 clusters in the weight matrix for the destination IP addresses (Input 2). Input 3 corresponds to the port numbers and Input 4 corresponds to the flow rates.

As for the SOM algorithm, each sample data will be mapped to neural network cells. We used 20 by 20 maps for examples in this section. However, the size of the neural network can be adjusted according to the scale of the dataset and the accuracy needed for the mapping. A hit map is then drawn to show the overall clustering based on all the input features. For the 3 source 2 destination IP addresses scenario, we manually setup only one flow-rate in the communication between all the source and destination hosts, and the hits map (Fig. 2.18) can basically show 6 clusters which means there are 6 flow pairs existing in the network.

Large scale studies (e.g., larger number of IP addresses) are further investigated. The results are shown in Figs. 2.19, 2.20, 2.21 and 2.22. As the number of pairs of communication becomes larger, visual inspection of both the weight and hits maps becomes more difficult. There are multiple ways to address this issue such as implementing a hierarchical SOM training algorithm [33], which will be discussed in Chapter. 3.

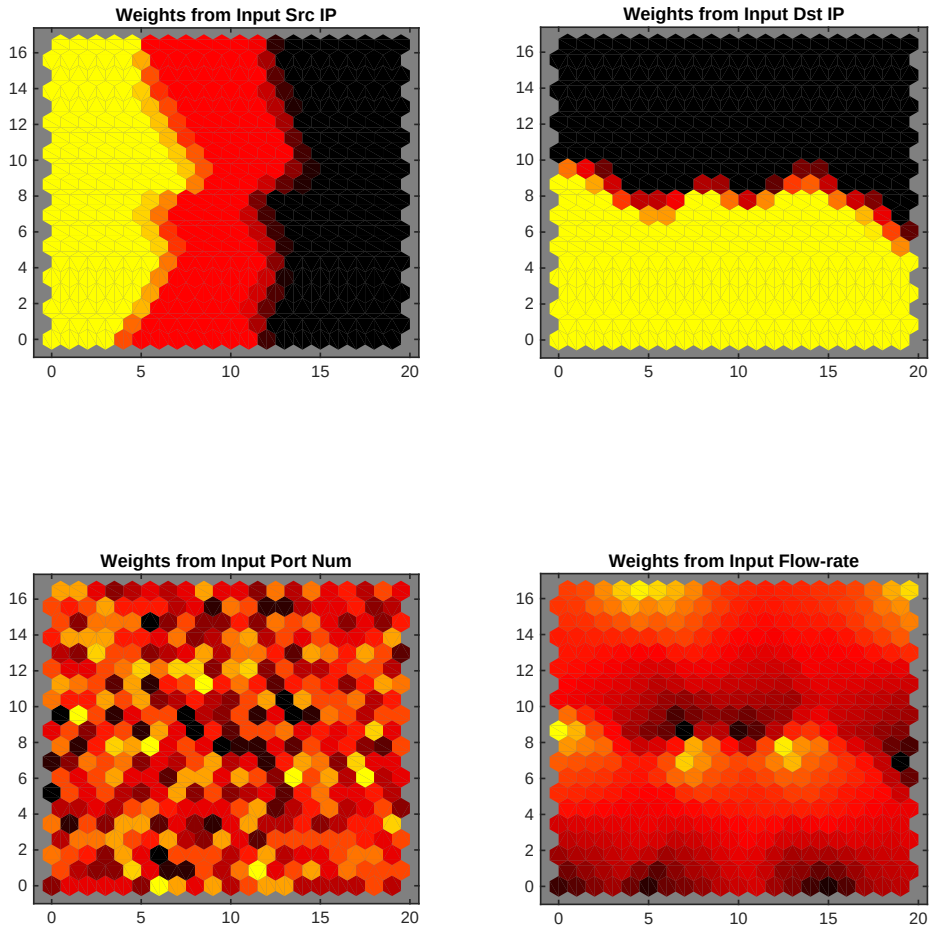


Figure 2.17: Weight Vector Map for SOM Training without Attackers (3 Source IPs, 2 Destination IPs), Input 1: Src IP, Input2: Dst IP, Input3: Port, Input4: Flow-rate, X Axis: Column Index for Neurons, Y Axis: Row Index for Neurons

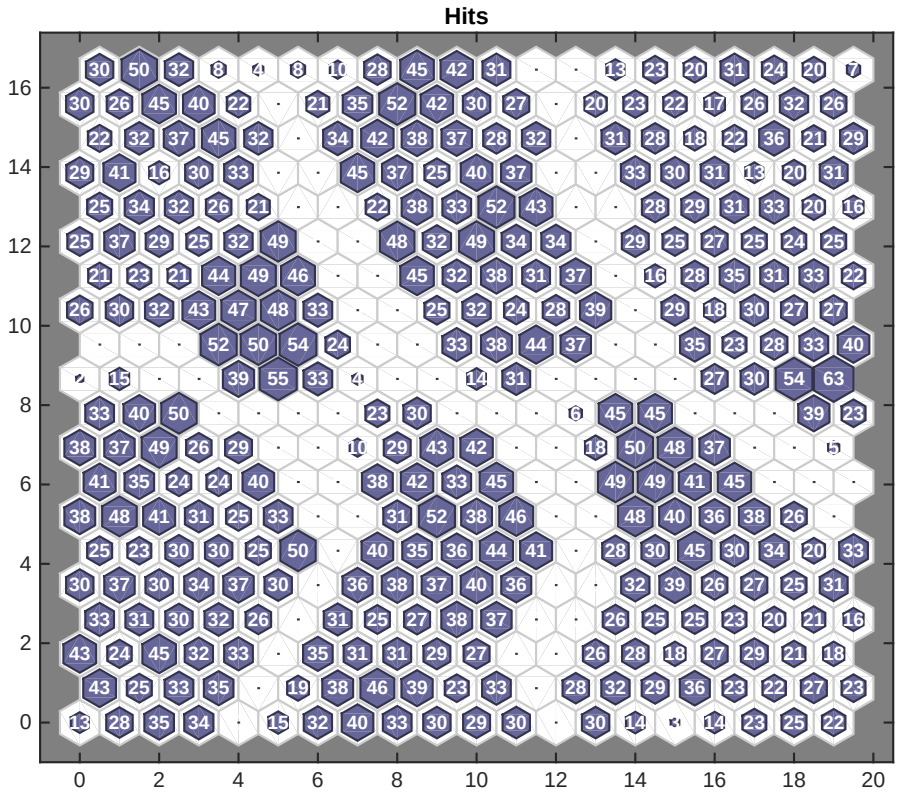


Figure 2.18: Hits (Cluster) Map for SOM Training without Attackers (3 Source IPs, 2 Destination IPs), 6 Clusters, X Axis: Column Index for Neurons, Y Axis: Row Index for Neurons

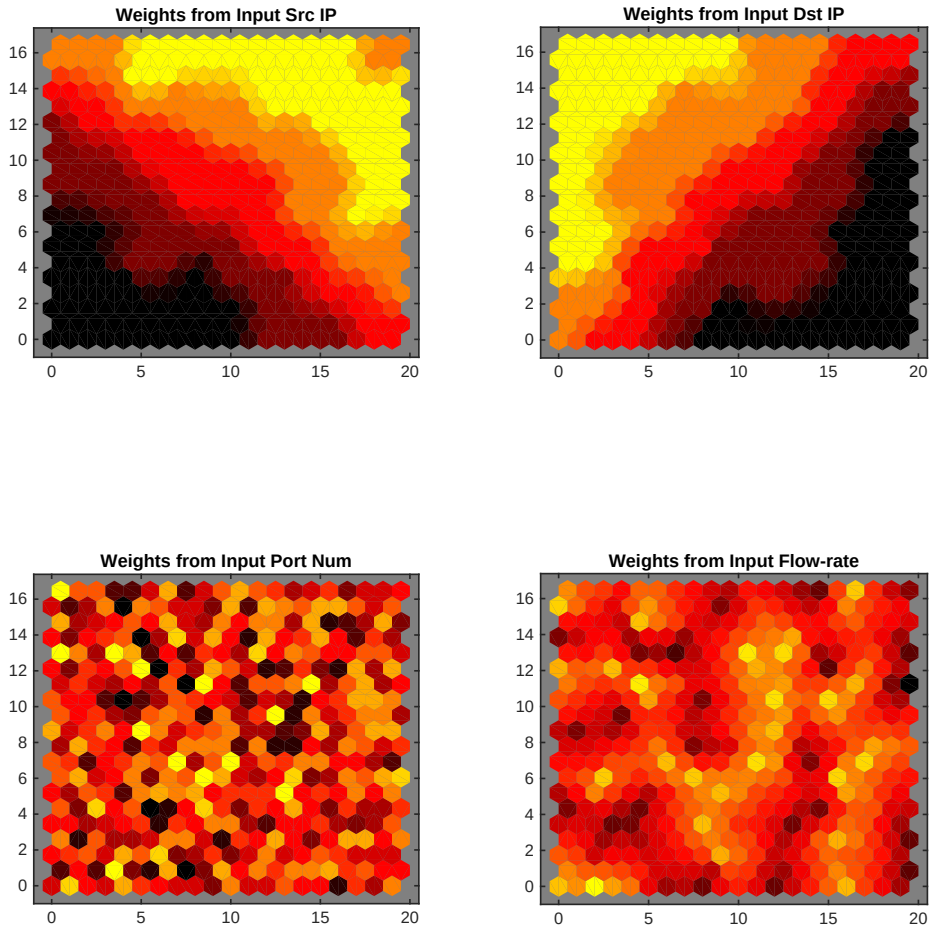


Figure 2.19: Weight Vector Map for SOM Training without Attackers (5 Source IPs, 5 Destination IPs), Input 1: Src IP, Input2: Dst IP, Input3: Port, Input4: Flow-rate, X Axis: Column Index for Neurons, Y Axis: Row Index for Neurons

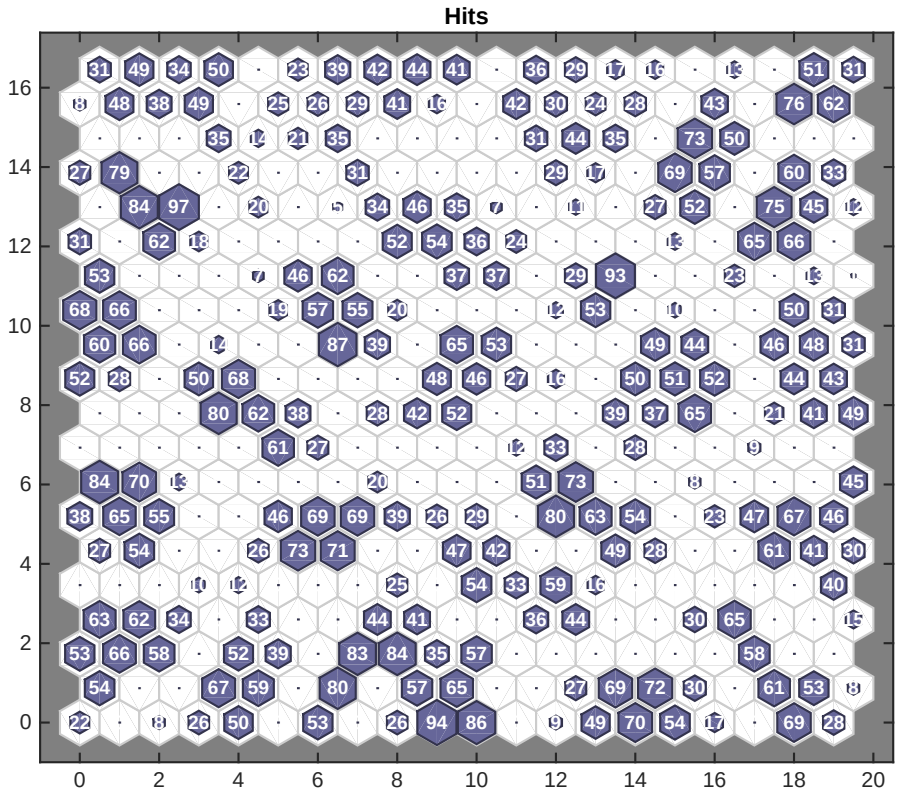


Figure 2.20: Hits Map for SOM Training without Attackers (5 Source IPs, 5 Destination IPs), 25 Clusters, X Axis: Column Index for Neurons, Y Axis: Row Index for Neurons

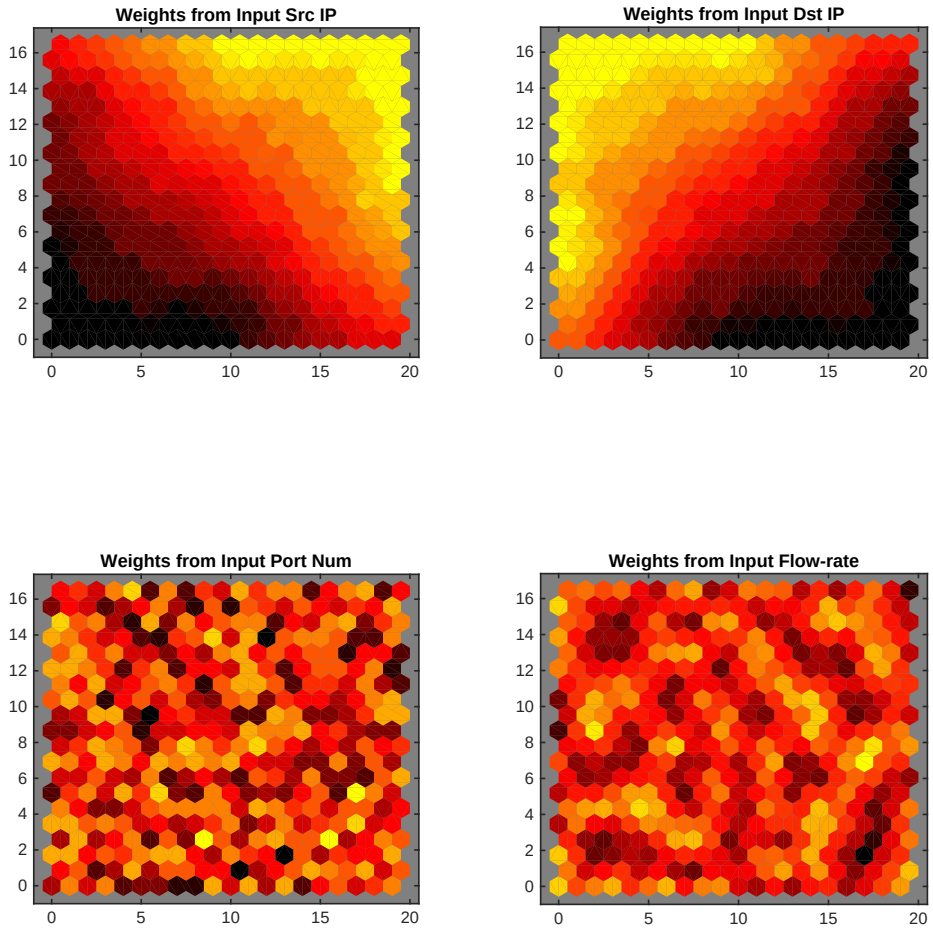


Figure 2.21: Weight Vector Map for SOM Training without Attackers (10 Source IPs, 10 Destination IPs), Input 1: Src IP, Input2: Dst IP, Input3: Port, Input4: Flow-rate, X Axis: Column Index for Neurons, Y Axis: Row Index for Neurons

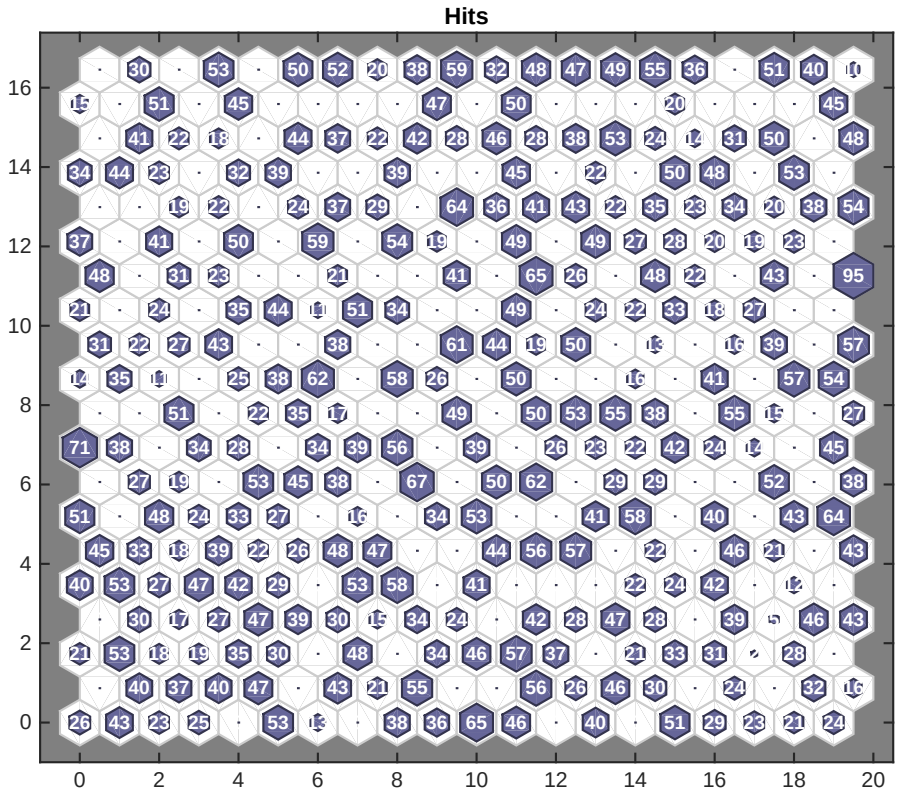


Figure 2.22: Hits Map for SOM Training without Attackers (10 Source IPs, 10 Destination IPs), 100 Clusters, X Axis: Column Index for Neurons, Y Axis: Row Index for Neurons

2.5.3 Advanced Maps with Coloured/Labeled Hit Regions

In order to pin-point the attacks in a SCADA network, we further utilize the SOM Toolbox 2.0 [34] to attach the labels to the data set. For example, Fig. 2.23 shows a coloured hits map for the 3 source, 2 destination IP addresses case. Although it is obvious to see the distinct clusters in this simple case, the colouring can be more powerful for the cases with larger number of clusters. This tool box also provides a range for colours next to the map that indicate a correspondence between a colour and a number. For example, in Fig. 2.25, the port number map is shown as all green indicating port 502 or in the Source IP map, yellow colour indicates x.x.x.40 address while blue node indicates x.x.x.10 address. This tool box also provides U-matrix and PC-projection as shown in Fig. 2.23. U-matrix is the unified distance matrix that provides a map based on the Euclidean distance between the neighboring nodes. PC-projection is the projection to principal-component space. In this work we focus on the utilization of weights and hit maps only.

Suppose the network is attacked by a new IP address, in order to pin-point the attacker, we can add labels to each of the sample data according to the IP addresses (or other features in the network). For example, based on the 3 source, 2 destination IP addresses case, if an attacker suddenly joins the SCADA network with a new IP address and starts sending packets to all devices in the network, not only can we tell that there is a change on the weight matrix for the source IP addresses (a new coloured cluster appears), but also we can label the neural cells with the source IP addresses as shown in Fig. 2.24 and Fig. 2.25. Then we can pin-point the IP for the attacker by comparing with the figures for the normal traffic.

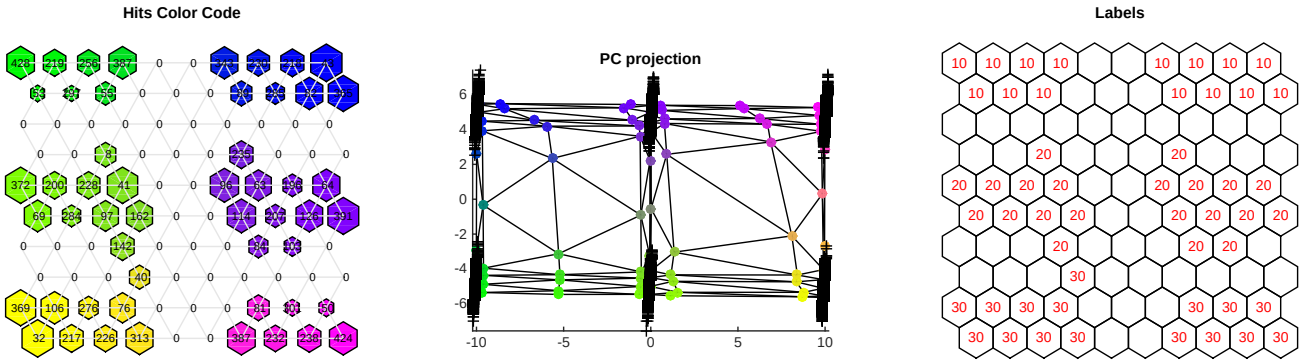


Figure 2.23: Coloured Hits Map for SOM Training without Attackers (3 Source IPs, 2 Destination IPs)

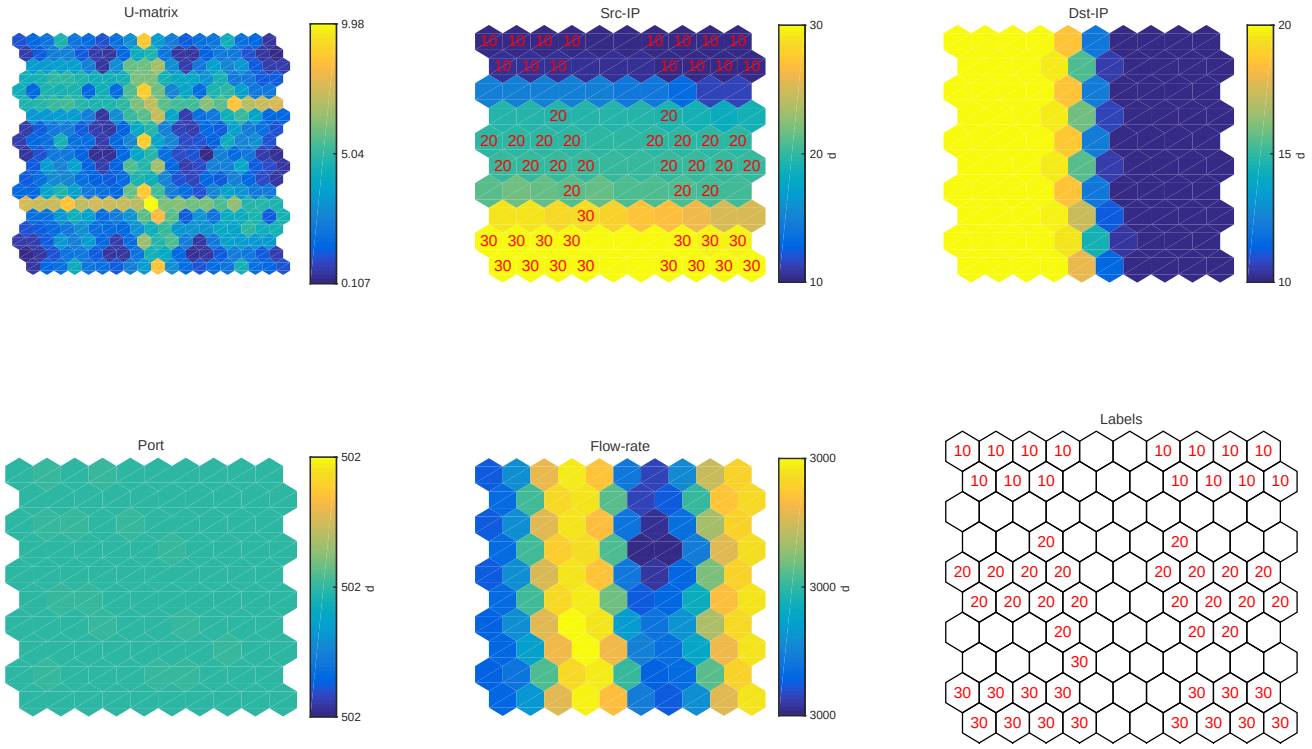


Figure 2.24: Src IP Labeled Hits Map for SOM Training without Attackers (3 Source IPs, 2 Destination IPs)

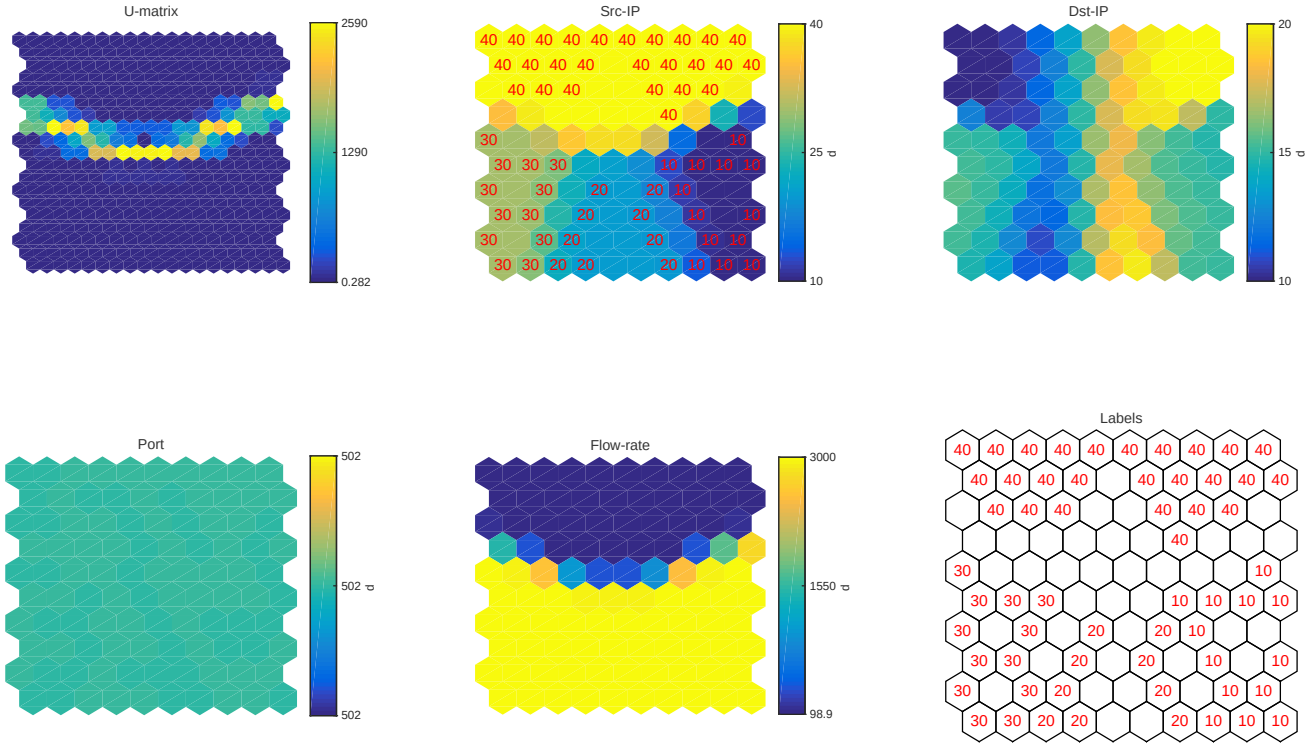


Figure 2.25: Src IP Labeled Hits Map for SOM Training with Attackers (4 Source IPs, 2 Destination IPs), Label 10 stands for 192.168.0.10 and so on.

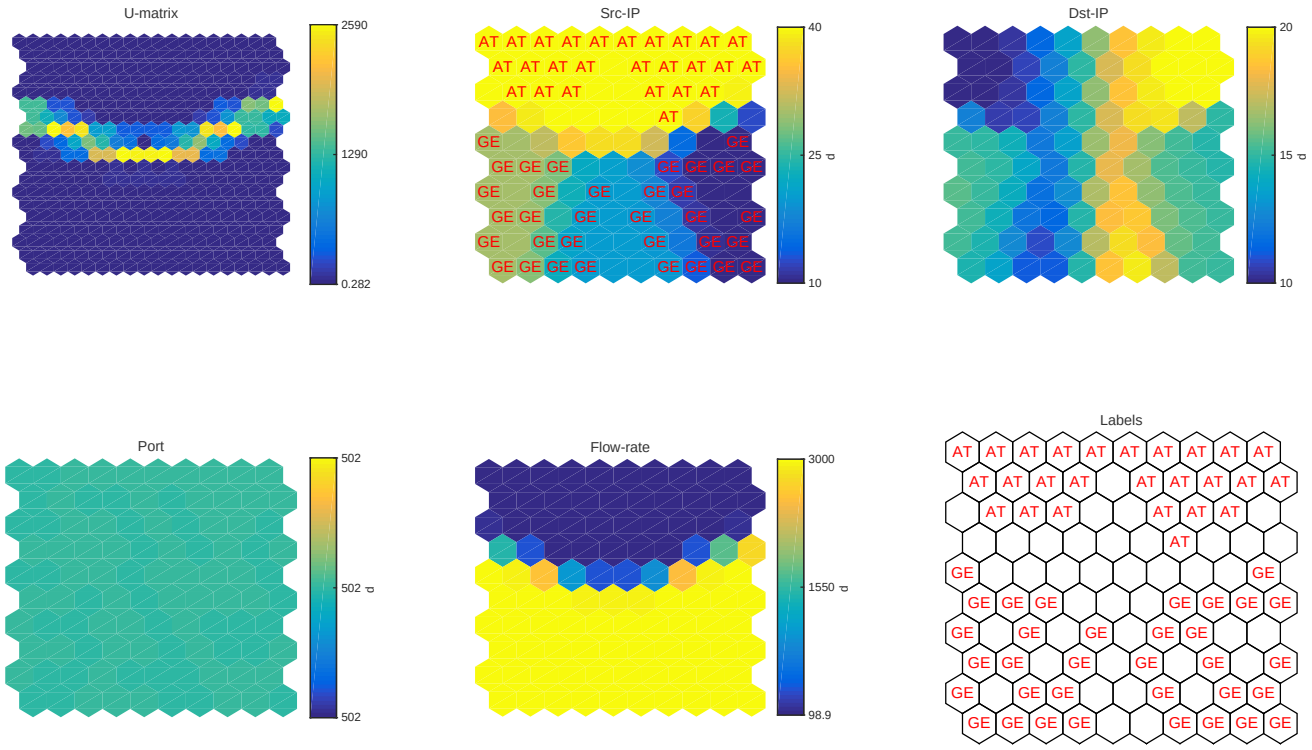


Figure 2.26: Labeled Hits Map for SOM Training with Attackers (4 Source IPs, 2 Destination IPs), Label 'GE' stands for general traffic and label 'AT' stands for attacks

2.6 Summary

In this chapter, we discussed the SOM-based algorithm with unsupervised learning to detect the traffic anomalies of SCADA network. SOM can be used as an efficient tool for visual inspection as the high dimensional dataset can be represented on two-dimensional maps. SOM can automatically learn about the training dataset and show the clustering results on the maps. If the new inputs cannot be mapped to any of the cells in the clusters on the trained maps, then the new inputs can be categorized as anomalies. However, as the training dataset becomes larger, it becomes hard to manually inspect the maps as more clusters will appear on the maps. Therefore, we need to find out a solution which can provide numerical criteria on the comparison between two different maps and this will be discussed in Chapter 4.

Chapter 3

An Enhanced and Quantitative Measurement on the Variations of Self-Organizing Maps

Self-Organizing Maps (SOM) can be used as an efficient tool for visualization. However, for human inspection it is not sufficient because it will be too subjective to tell the difference between two SOM maps. Therefore, we search for the solution which can give an deterministic measurement on the different appearance of SOM maps. In this section, we will discuss the methods which are applied after the training with SOM algorithm. A Gaussian Mixture Model (GMM) is used to obtain the kernels/cluster centers from the SOM map, and to compare and compute the distance between the original trained SOM map and the new testing SOM map with potential anomaly traffic, the Kullback-Leibler Divergence (KLD) is utilized.

3.1 Introduction of GMM Method

Gaussian mixture model (GMM) belongs to the theory of probabilistic models. It assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. For example, for a multivariate distribution you can model a vector of parameters using a Gaussian mixture model prior distribution on the vector of estimates given by [35]:

$$p(\theta) = \sum_{i=1}^{K} \phi_i N(\mu_i, \Sigma_i). \tag{3.1}$$

where K is the number of Guassian components, the i^{th} vector component is characterized by Gaussian distributions with weights ϕ_i. Each of the Gaussian distributions has mean μ_i and covariance matrices Σ_i. Figures 3.1 and 3.2 show a simple example of how the GMM algorithm works. We take advantage of the "Fisher's Iris Data Set" which is provided in the software of Matlab for tests on clustering.

After applying the GMM algorithm, we can see three clusters or three gaussian distributions in Fig. 3.2, with the crosses in the centers of the clusters.

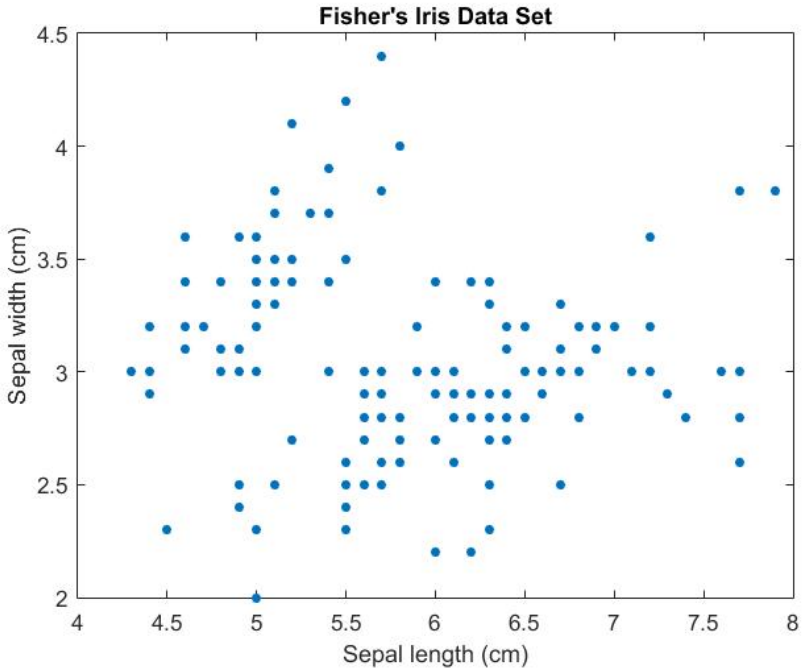


Figure 3.1: Using GMM Toolbox in Matlab

The SOM Toolbox 2.0[34] natively supports the GMM algorithm and we can directly apply GMM after the SOM training. The parameters of the Gaussians are determined using the Expectation-Maximization (EM) algorithm [36]. In kernel estimation, diagonal forms of the covariance matrix of each cluster are often used, because inside a cluster the variables can be assumed to be independent. As mentioned in [37], Netlab software package by Nabney and Bishop [38] was used in the GMM simulations. In order to estimate the probability density using the SOM, the toolbox helps

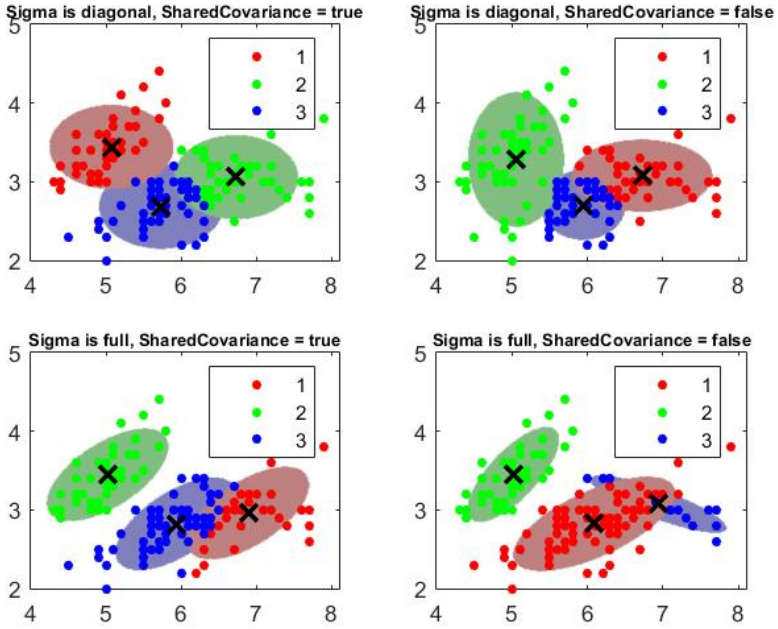


Figure 3.2: Simple GMM Example in Matlab

to build a RKDE model using the map model vectors as kernel centers [37]. The prior probability for map unit i is ratio

$$P(i) = \frac{\sum(x_n \in V_i)}{\sum x_n}, n = 1, ..., M, \tag{3.2}$$

where V_i is the Voronoi set of unit i (set of data vectors whose closest model vector is m_i) and M is the number of samples [37].

We manually make a sample dataset with three Gaussian distribution (different mean value but the same variance). After the SOM training, we further apply the GMM algorithm and Fig. 3.3 shows the output, which is named as the GMM Kernel Probability. In more details, each of the cell on the trained self-organizing map has one value referring to the kernel probability, and the density functions $p(x|i)$ for each map unit are estimated using corresponding Voronoi sets V_i. As shown in Fig. 3.3, the top largest numbers or the brightest colors are indicating the centers of the clusters/Gaussian distributions.

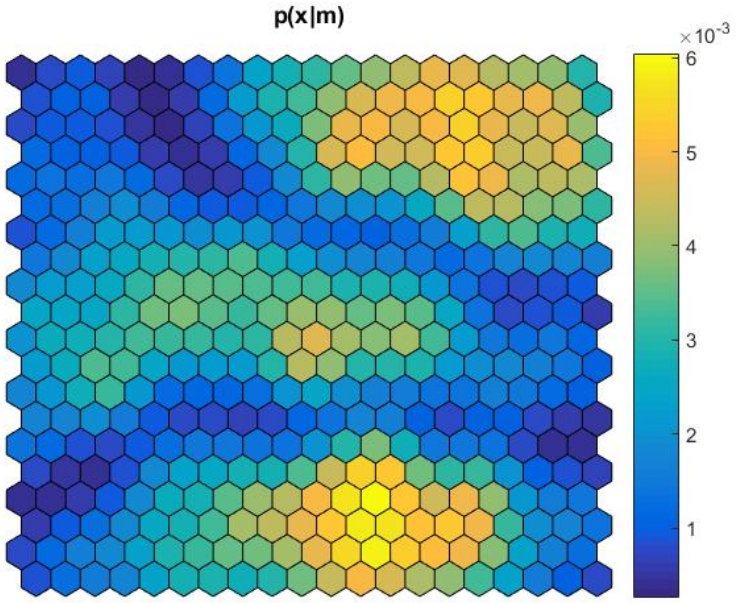


Figure 3.3: GMM Kernel Probability

3.1.1 Distance Computing with Kullback-Leibler Divergence

With the help of the GMM algorithm a kernel probability based on the trained SOM can be obtained, which can be used as the input to compute the Kullback-Leibler Divergence (KLD). The reason why we need to apply the KLD algorithm is because the Kullback-Leibler Divergence[39] is generally used to compare two distributions and show the difference with numerical analysis. For two discrete probability distributions P and Q, the KLD of Q from P is defined as:

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)}. \tag{3.3}$$

For example, if we have two gaussian distributions with different means and variances, the KLD number will show how large the difference can be. As shown in Fig. 3.4, when the means of the two testing gaussian distribution become closer, the KLD number will be smaller. For two exactly identical gaussian distributions, the KLD number would be zero as shown in Fig. 3.5.

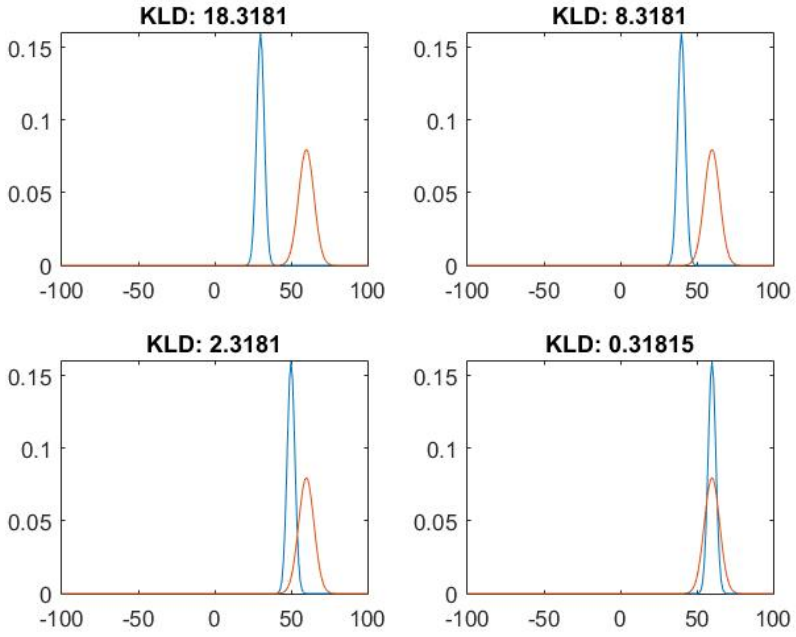


Figure 3.4: Computing the Kullback-Leibler Divergence for Comparison, X Axis: Values from Distribution, Y Axis: Probability Density Function (PDF)

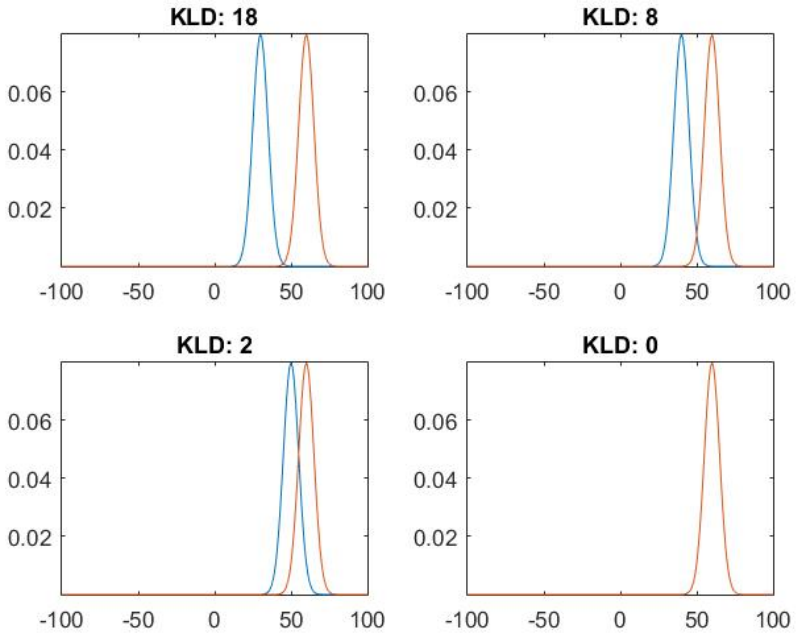


Figure 3.5: Computing the Kullback-Leibler Divergence for Comparison, X Axis: Values from Distribution, Y Axis: Probability Density Function (PDF)

3.1.2 Time Lapse Analysis

The analysis so far has focused on a single data set (with or without attacks) in order to evaluate the performance of a SOM method. In a practical world these tools will be used on a real-time basis. Therefore this section focuses on the SOM anomaly detection as a data-stream application, in which, it is assumed that continuous monitoring data is available as a function of time. An OMNET++ simulation of 500 seconds is conducted with a time sequence of traffic flows as shown in Fig. 3.6. A single attack occurs at 300 second mark and lasts for about 30 seconds (about 20000 packets). A 20 second time-lapse data is fed to a SOM in order to understand how the SOM analysis would look like. The results of this analysis are shown in Fig. 3.7 and 3.8 without any attacks; Fig. 3.9 and 3.10 show with attacks. The attack stopped at about the 330 second mark and so Fig. 3.11 represents the data again without any attack. This time lapse indicates a clear change in the flow rate and source address maps. The visual maps coupled with the range values (shown next to each map) give a very good indication of the significant change in the traffic behavior.

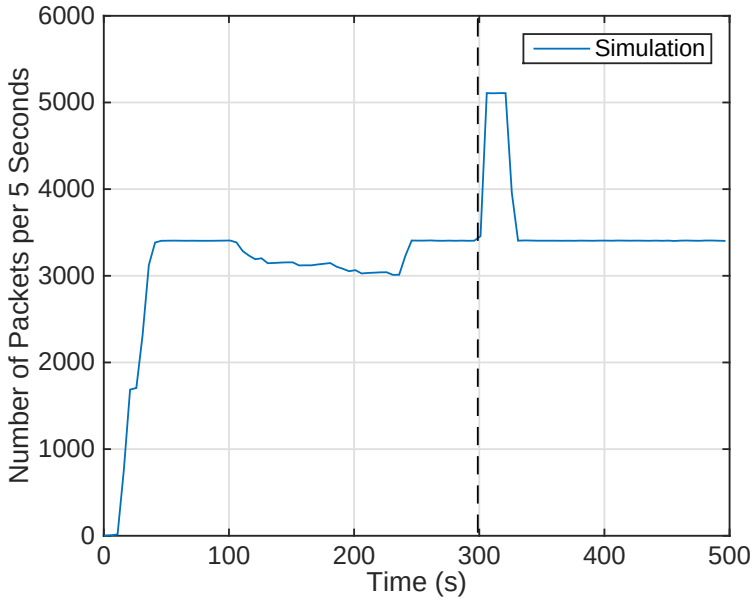


Figure 3.6: Time Sequence IO Graph from OMNeT++ Simulation

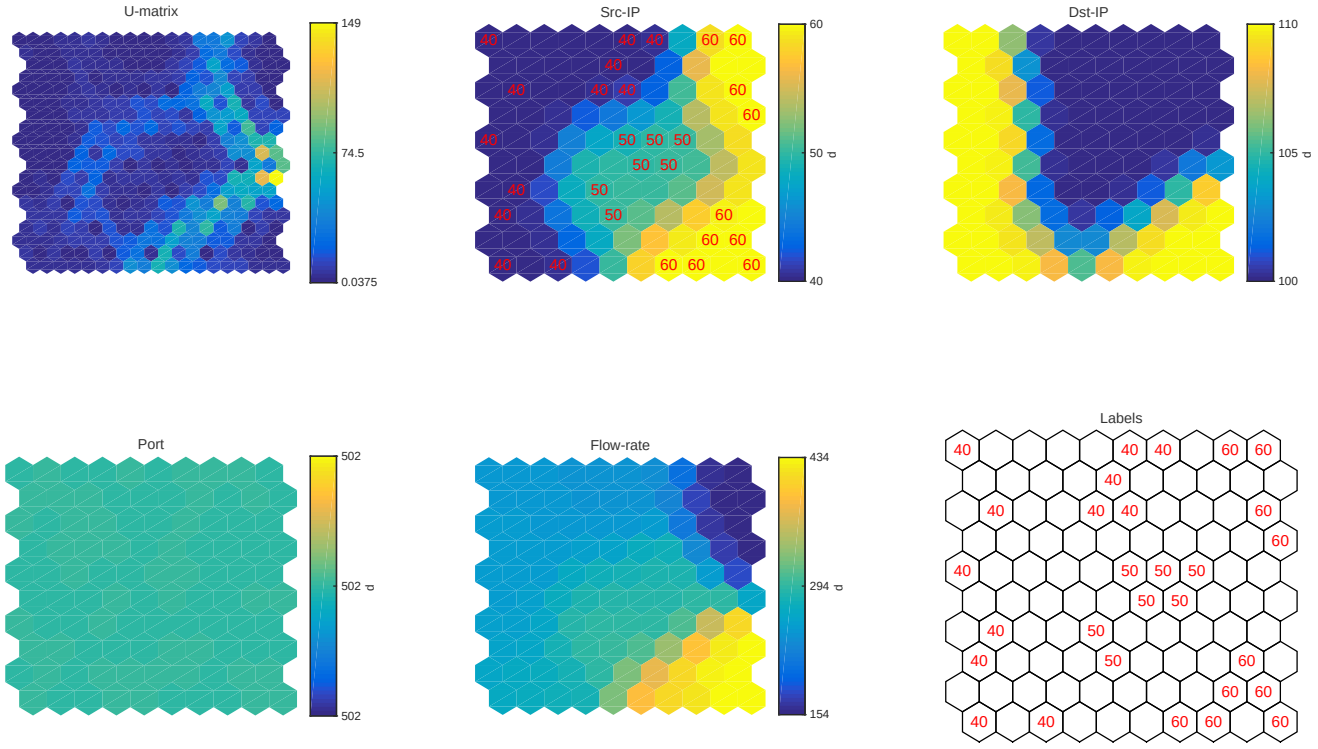


Figure 3.7: SOM Training without Attack at 250-270 second mark (3 Source IPs, 2 Destination IPs)

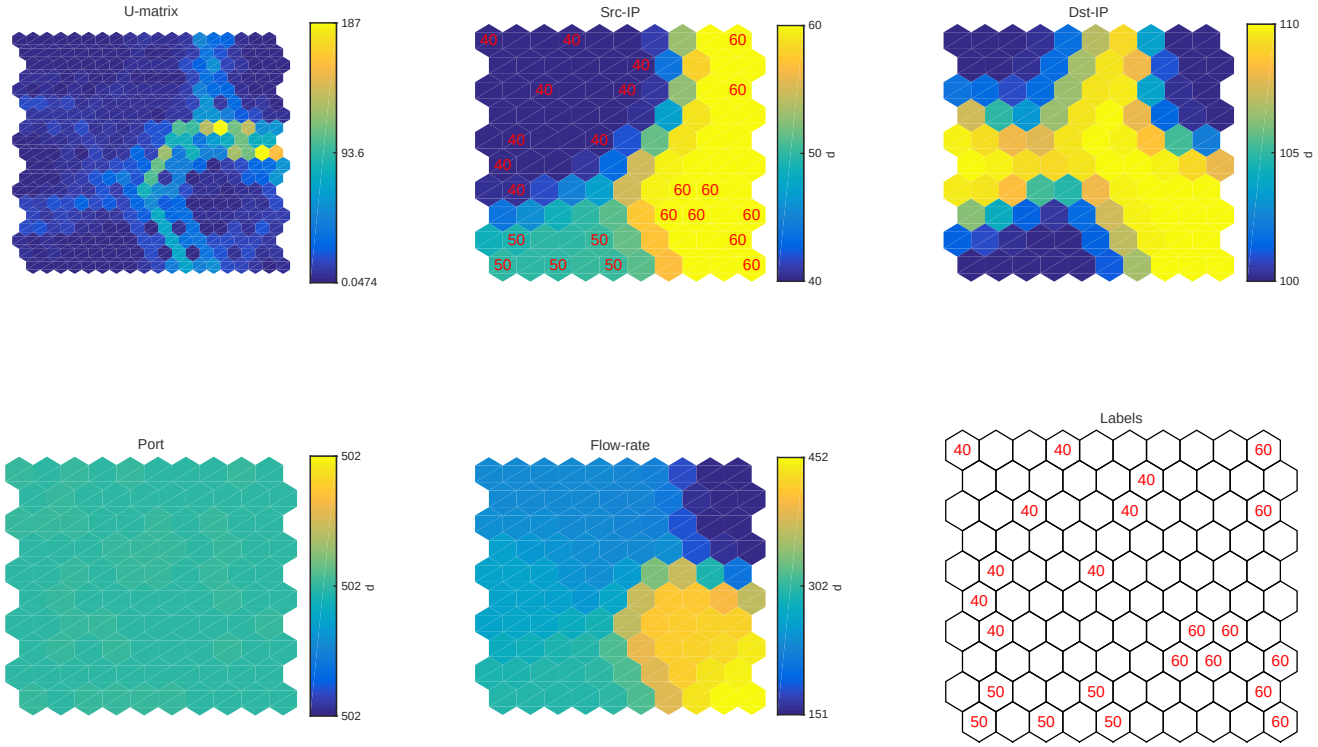


Figure 3.8: SOM Training without Attack at 270-290 second mark (3 Source IPs, 2 Destination IPs)

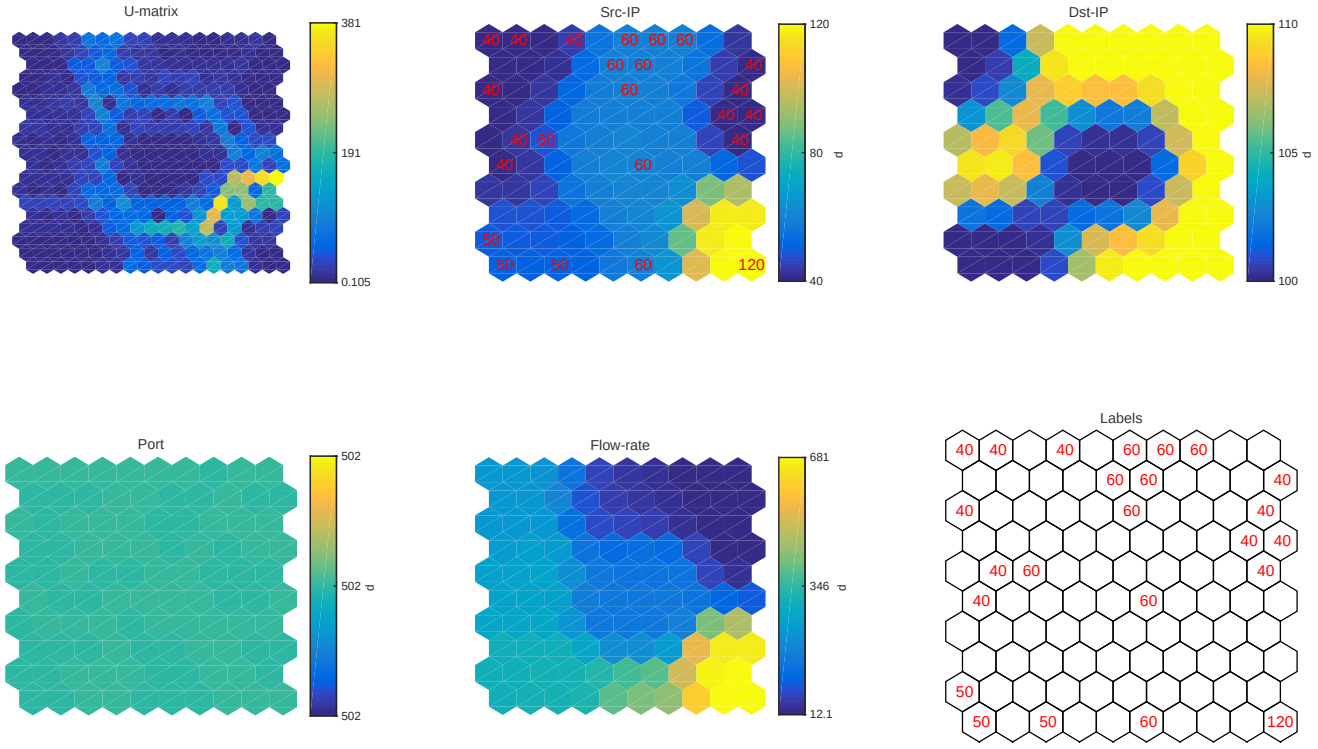


Figure 3.9: SOM Training with Attack at 290-310 second mark (4 Source IPs, 2 Destination IPs)

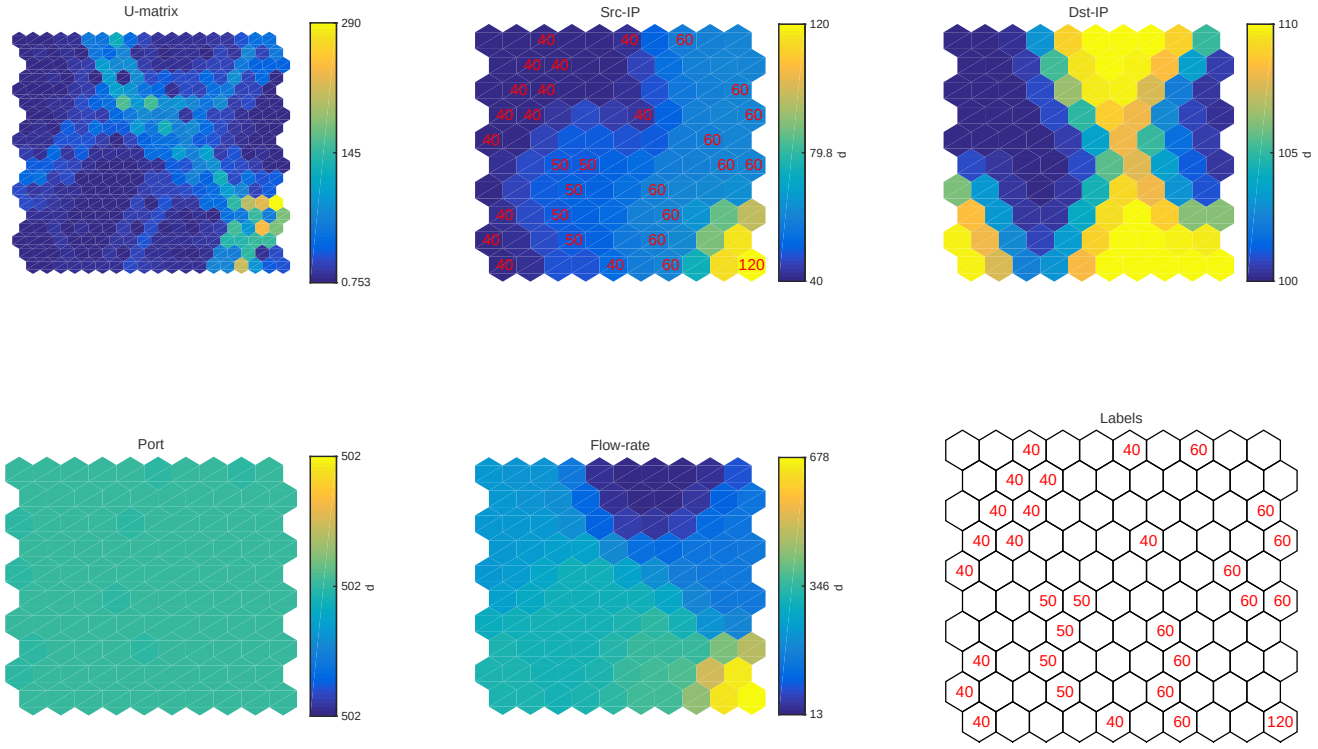


Figure 3.10: SOM Training with Attack at 310-330 second mark (4 Source IPs, 2 Destination IPs)

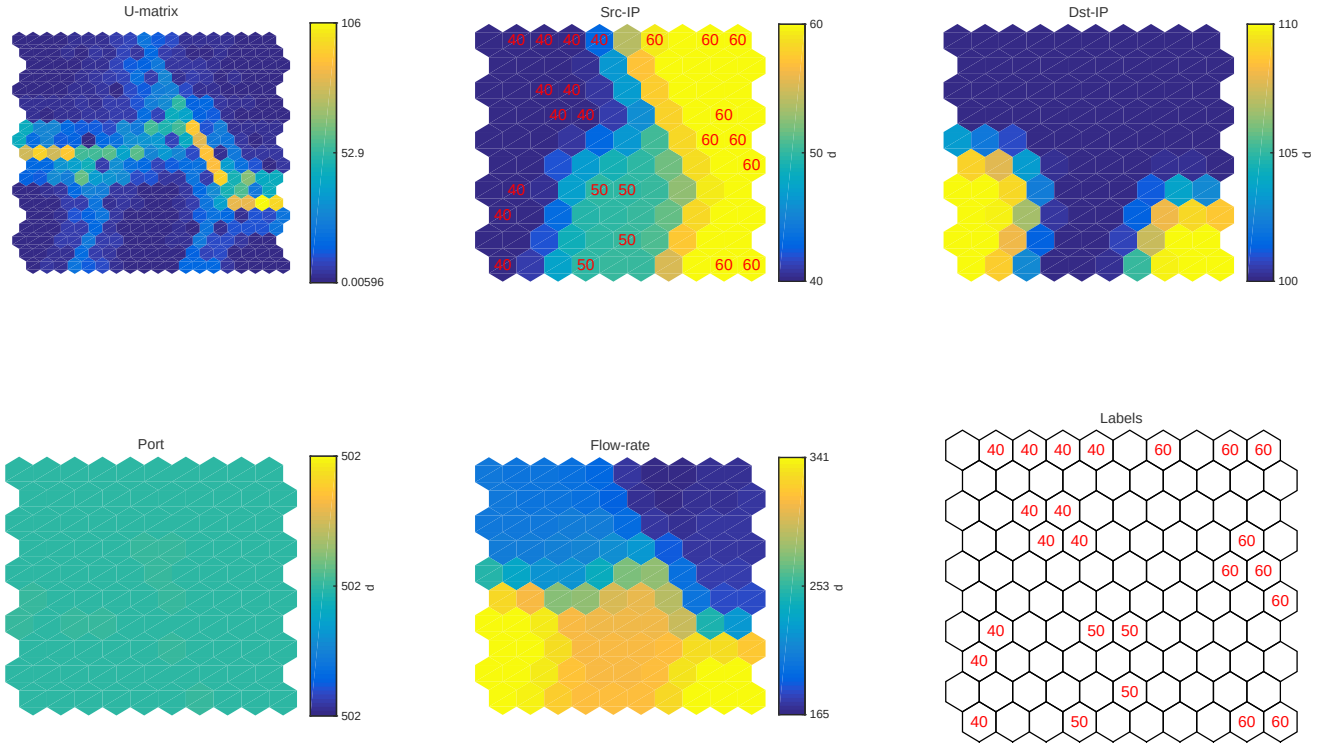


Figure 3.11: SOM Training without Attack at 330-350 second mark (3 Source IPs, 2 Destination IPs)

3.1.3 Apply GMM and KLD to the Results of SOM

In order to obtain a threshold for deciding whether an anomaly happens or not in the SCADA network to aid human visual inspection, further analysis based on the GMM and KLD algorithms are applied to the training results of SOM.

To verify our approach, a new attack scenario is implemented in the simulation. As shown in Fig. 3.12, we set up the DoS attacker in the network to start attacking at time 150s (1500 in the simulation), and end around 180s (1800 in the simulation). Thus we expect to see a sudden increase in the flow-rate at 150s and then a decrease at around 180s. This is the attack scenario and we take the flow-rate and port number as two features for the SOM training inputs. This time we take a sliding window for the SOM training, i.e., every 50s (500 length in the simulation) as a separate dataset, and see what the trained self-organizing maps look like.

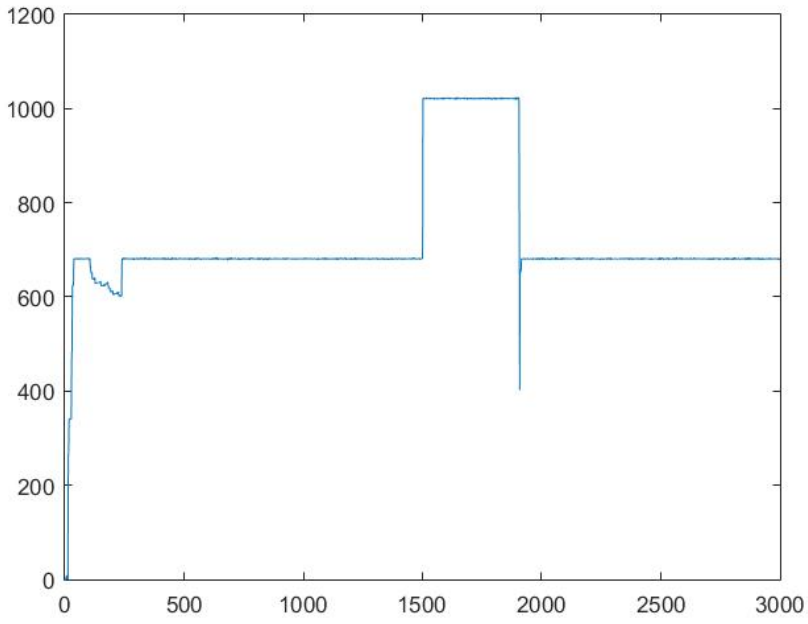


Figure 3.12: A New Attack Scenario in SCADASim

Figures 3.13, 3.14, 3.15 and 3.16 shows the results after SOM training. Comparing the SOM training results between the training window 500 to 1000 (Fig. 3.13) and the training window 1000 to 1500 (Fig. 3.14), the SOM results could be in different pattern, however if we take a look at the kernel probability which is generated with

the GMM algorithm, there is no big change in the figure. This shows the consistency with the Fig. 3.12, in which the flow-rate keeps stable.

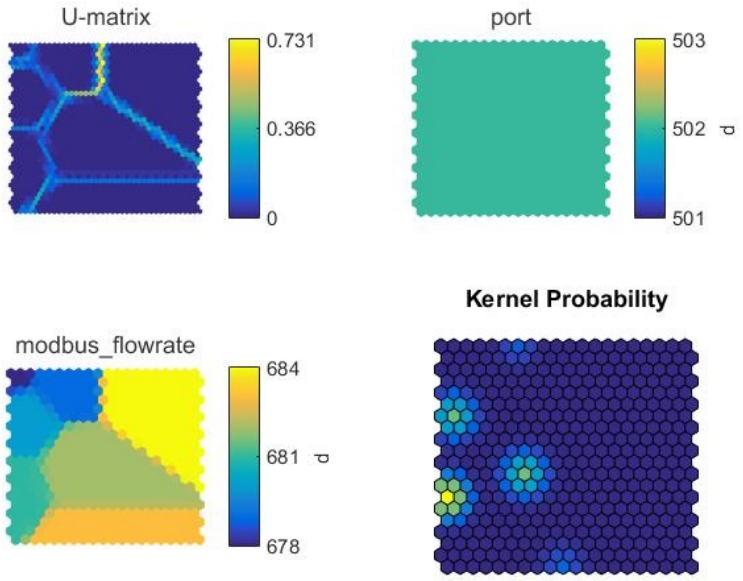


Figure 3.13: SOM Training for New Attack Scenario (500 to 1000)

Comparing Fig. 3.14 and Fig. 3.15, not only the SOM training results, but also the kernel probability changes as the attack starts at 150s (1500 in the simulation) which causes the increase in the flow-rate. A similar conclusion can be inferred by comparing Fig. 3.15 and Fig. 3.16.

We further applied the KLD method based on the SOM training results and Fig. 3.17 shows the effectiveness of our approach, in which we can see a small number of KLD when the flow-rate is stable and a large number of KLD if the flow-rate changes. This is the main focus and contribution in this dissertation on adding numerical methods to SOM algorithms, with anomaly detection as example domain.

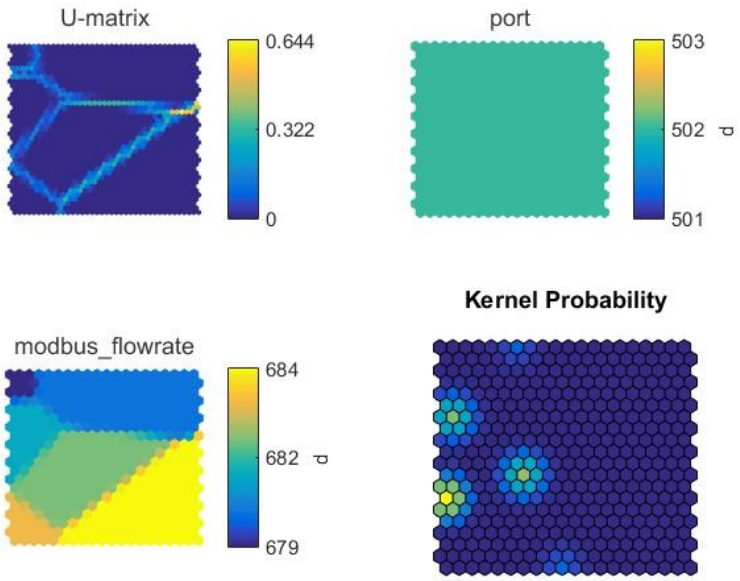


Figure 3.14: SOM Training for New Attack Scenario (1000 to 1500)

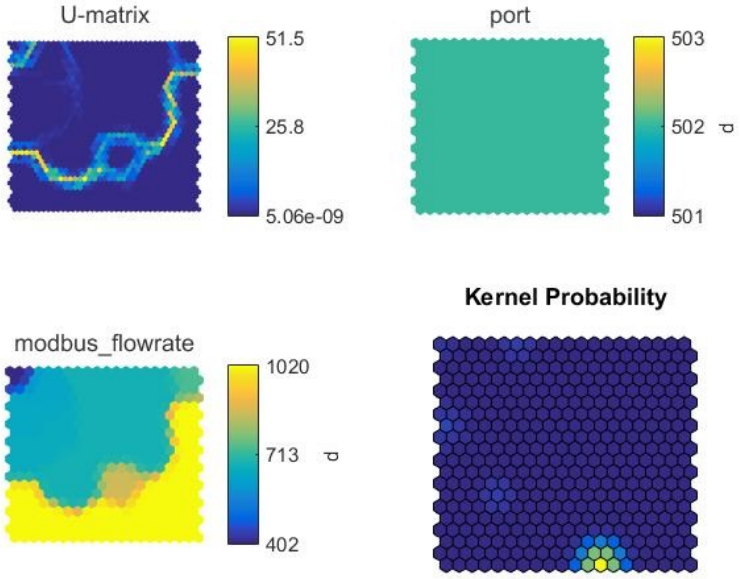


Figure 3.15: SOM Training for New Attack Scenario (1500 to 2000)

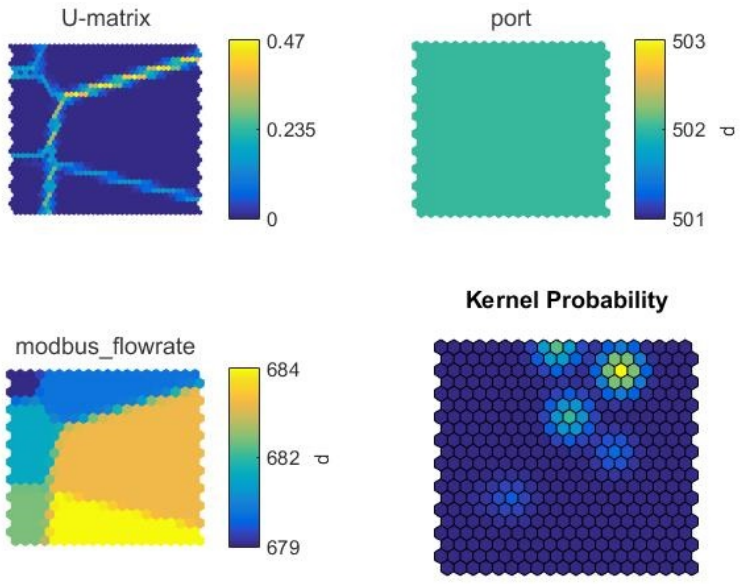


Figure 3.16: SOM Training for New Attack Scenario (2000 to 2500)

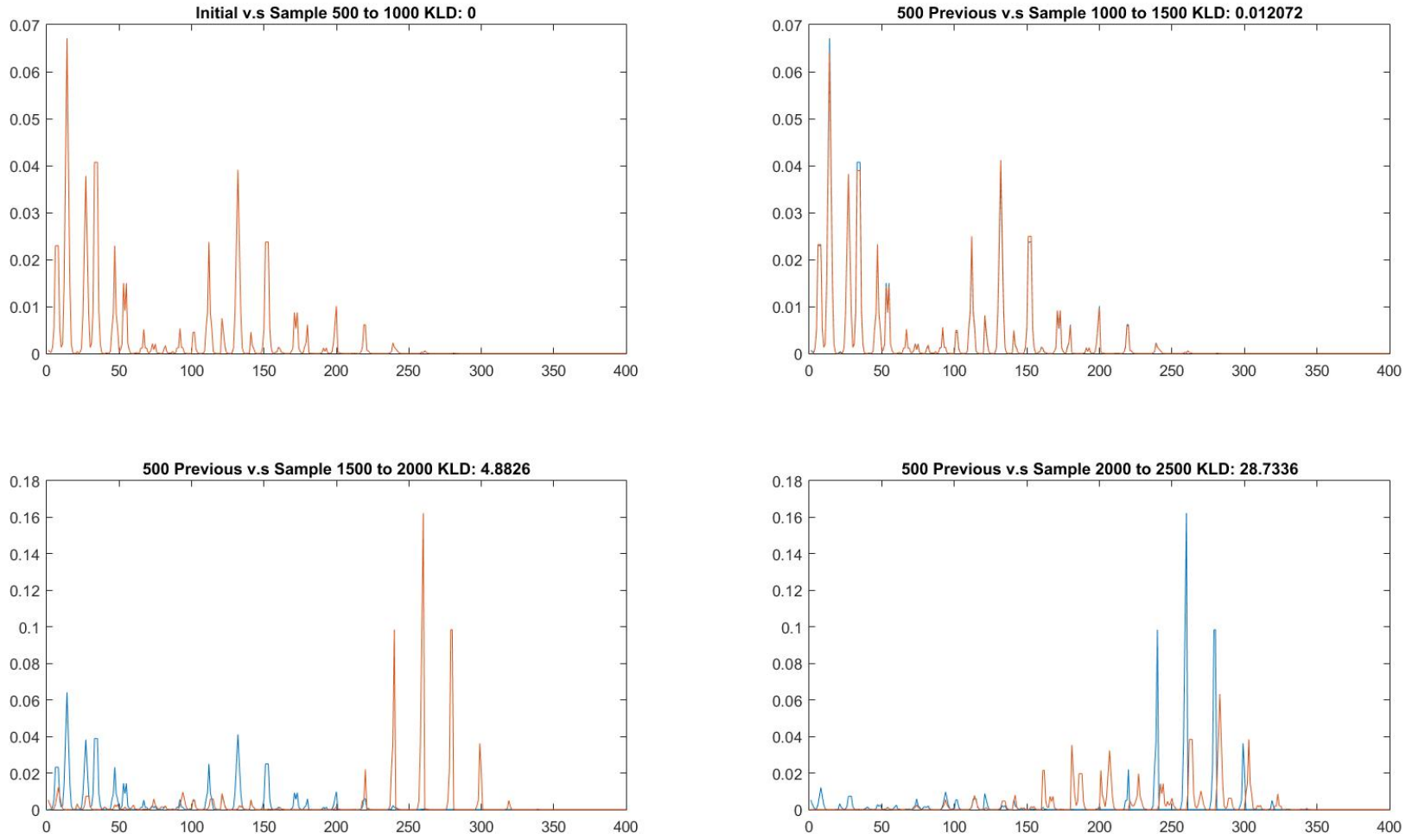


Figure 3.17: KLD for New Attack Scenario

3.2 Results with CTU-13 Dataset

In this section, we apply our SOM based approach on another dataset, the CTU-13 dataset, which was captured in the CTU University. It consists in a group of 13 different malware captures done in a real network environment, including Botnet, Normal and Background traffic. It is a labeled dataset in a flow by flow basis. As the CTU-13 dataset is large and contains multiple parts with different attacking scenarios, in our experiments we utilize the part 7 only (114078 records in total, including anomalies). More details can be checked in the online documents of the CTU dataset [40].

After the SOM training with the pure normal traffic, we also applied the GMM and KLD analysis on top of the training map. The results are shown as follows.

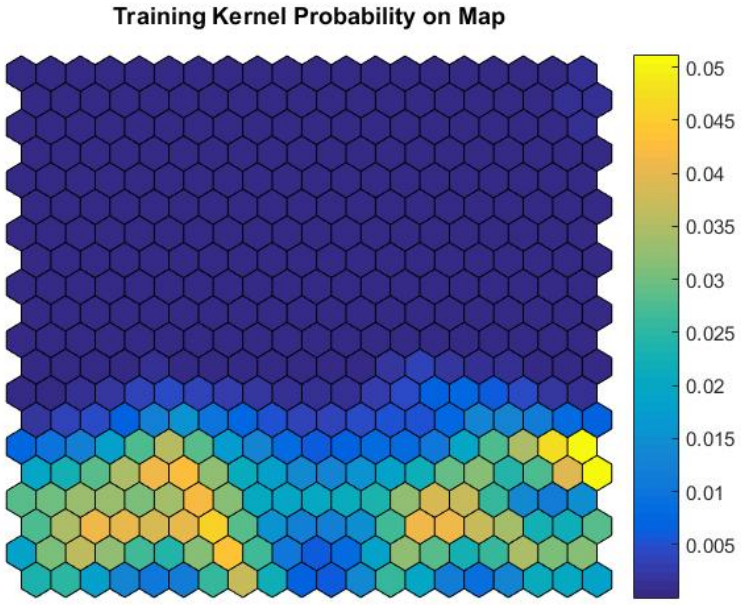


Figure 3.18: GMM Kernel Probability for the CTU Dataset (without Anomaly)

Figures 3.18 and 3.19 show the GMM results after the SOM training for each of datasets with pure normal traffic and with anomaly attacks. We can see the difference on the locations of the kernels with GMM outputs. Also as shown in Figs 3.20 and 3.21, if we apply the KLD method we can see the KLD number becomes larger when comparing the normal traffic and the anomaly traffic.

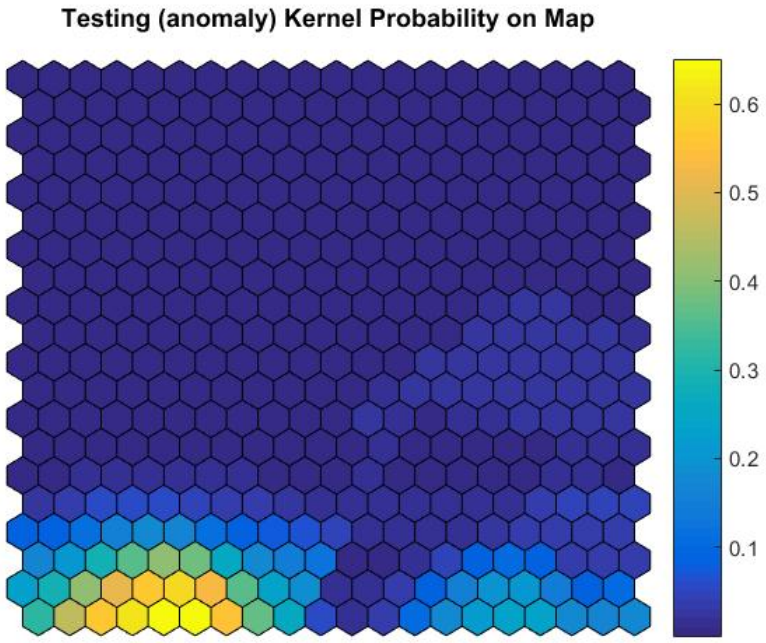


Figure 3.19: GMM Kernel Probability for the CTU Dataset (Anomaly Only)

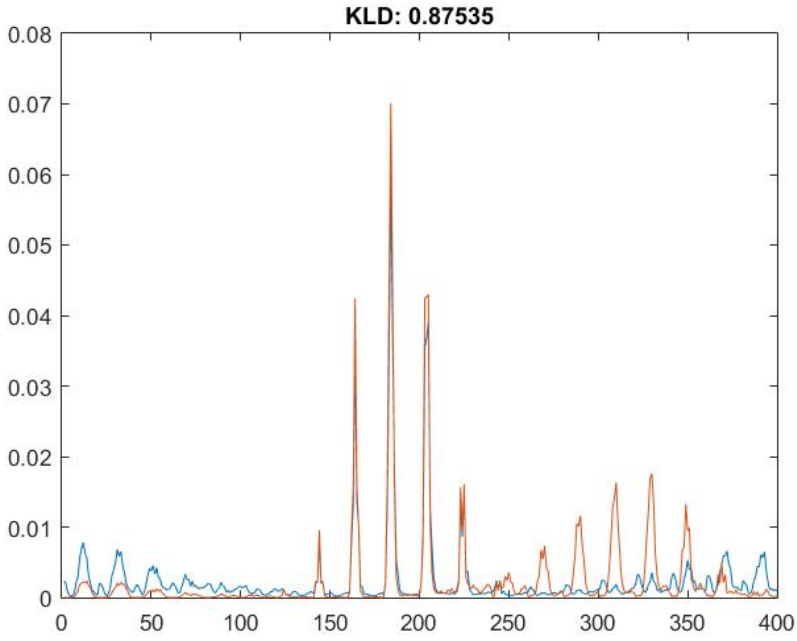


Figure 3.20: KLD for the Normal Traffic in the CTU Dataset

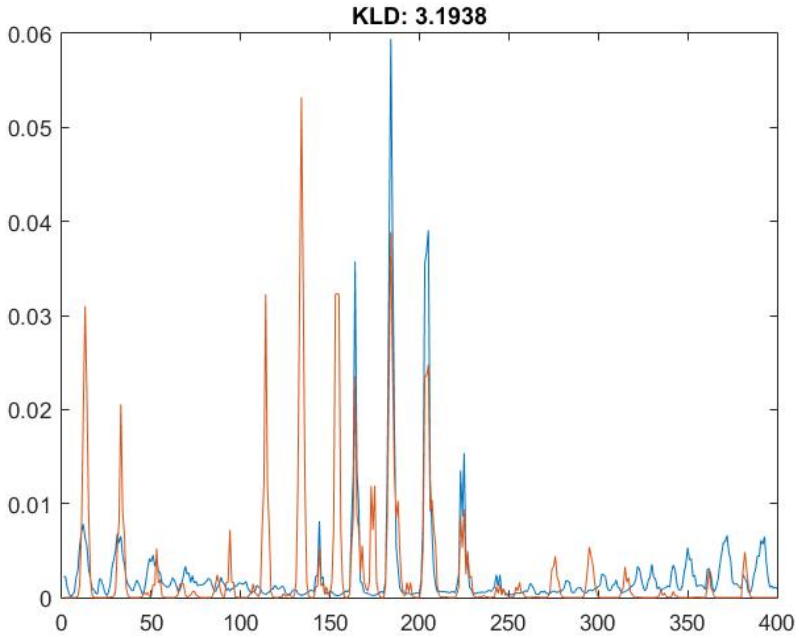


Figure 3.21: KLD for the Anomaly in the CTU Dataset

In the next experiment, in order to decide which number could be used as the threshold for anomaly detection, we take 4 small random samples, each has 1000 sample points from the original training dataset with only the normal traffic, and then compute the KLD number when comparing to the whole training dataset. In Fig. 3.22 it shows an average KLD number 0.0145. However, this number cannot be directly used as the threshold because the average KLD number also depends on the length of the random samples. For example, if each time we only take 500 sample points from the training dataset, the average KLD number becomes larger as shown in Fig. 3.23.

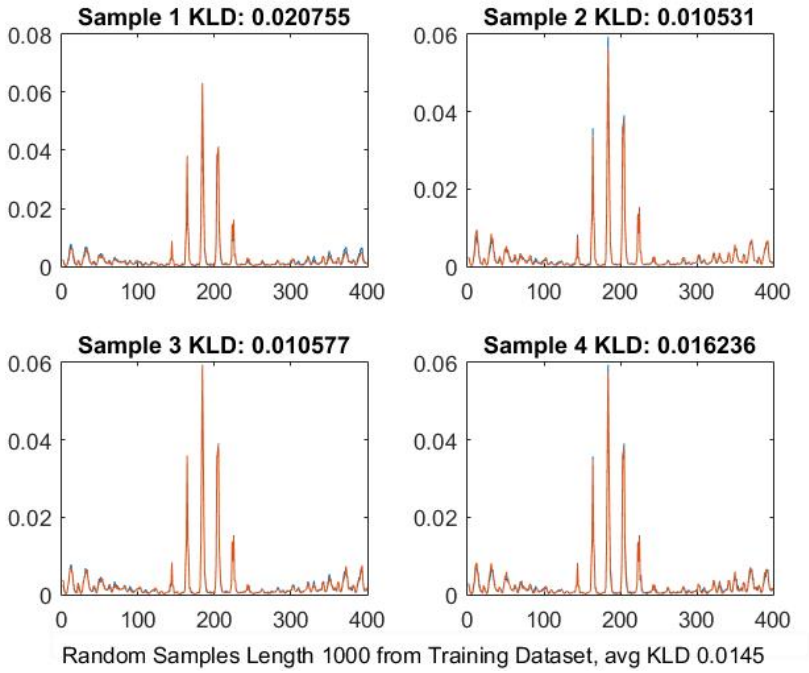


Figure 3.22: KLD for Random Sampling (1000) from the CTU Dataset

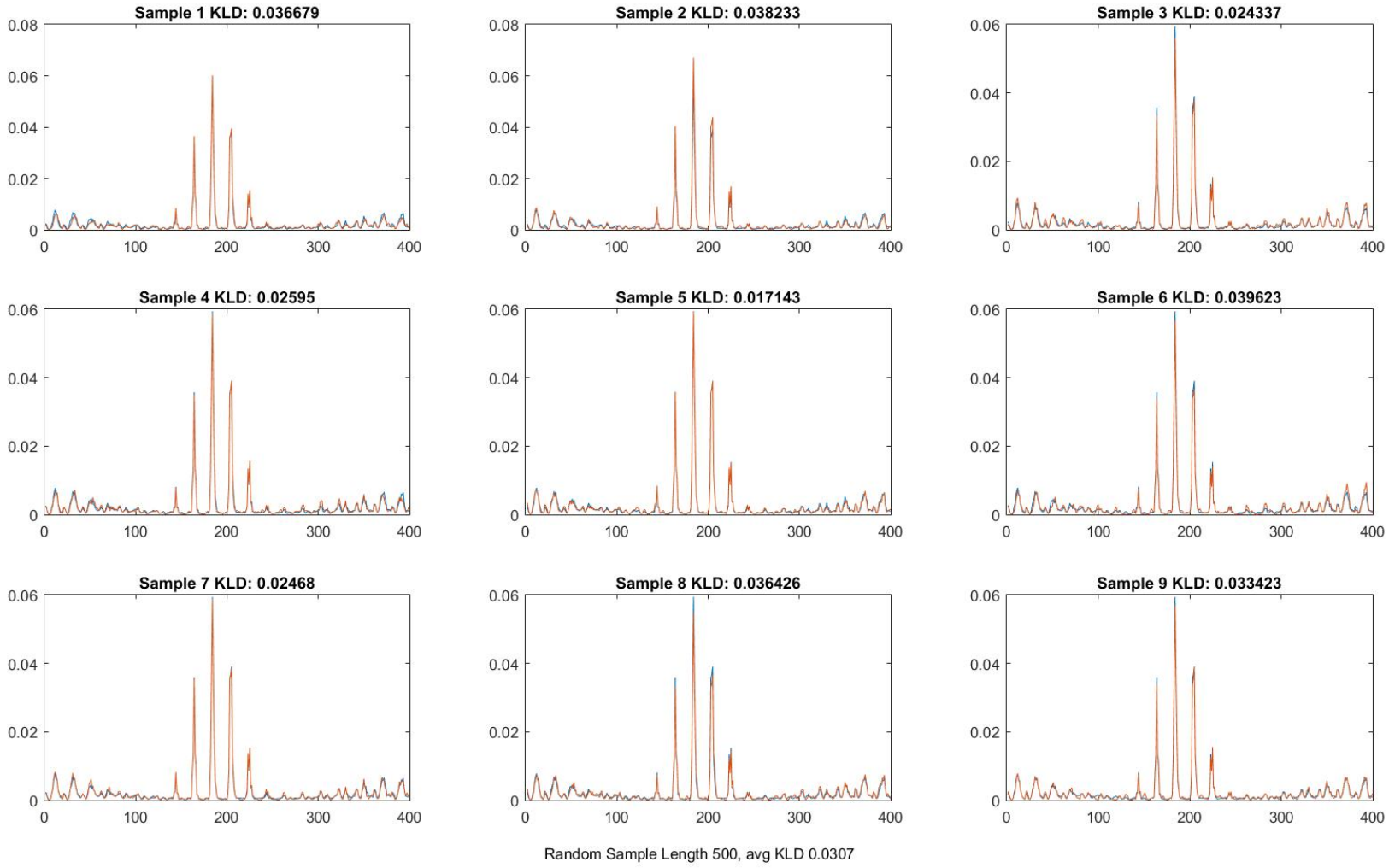


Figure 3.23: KLD for Random Sampling (500) from the CTU Dataset

Therefore, we further implement more experiments to see how the KLD number changes with different length of random samples. As shown in Fig. 3.24, there is a significant increase if the length of the random samples becomes less than 200. However, we can still find the KLD number less than 1.0, which is smaller than the KLD number when comparing the dataset with normal traffic and the one with anomaly attacks. In our case, we suggest to take 0.05 as the threshold for the alert of anomaly. This is because the curve becomes gentle when the "Sample Length" is greater than 300 as shown in Fig. 3.24. This is a completely ad hoc method of finding a threshold and future work needs to be done to develop a more rigorous approach.

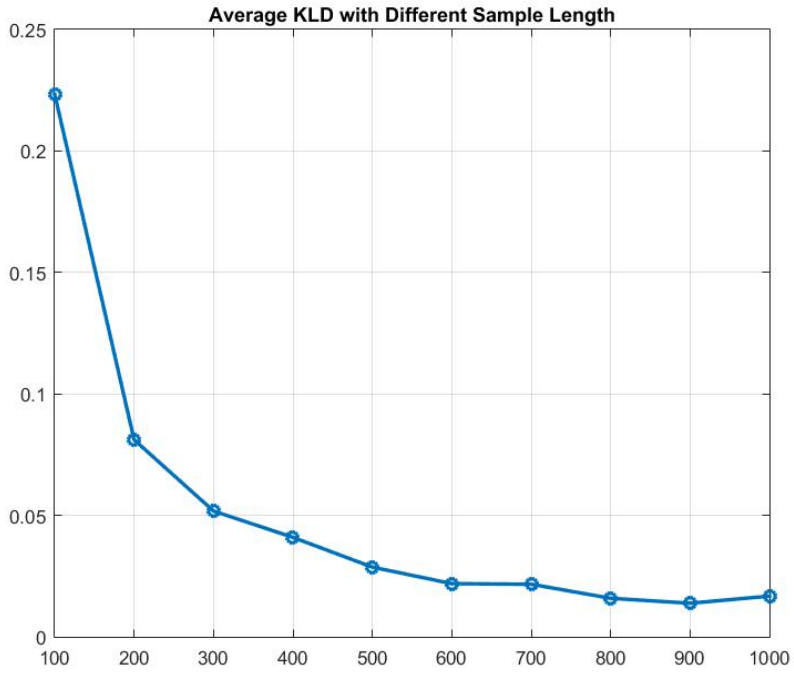


Figure 3.24: Average KLD v.s Different Length of Random Sampling from the CTU Dataset

Table 3.1 shows that at anomaly mixture-rates (the percentage of the anomalous inputs in the training dataset) of 12% or above, this approach has an accuracy of 100% in our case study from the CTU-13 dataset with the suggested threshold value 0.05. Moreover, if we adjust the threshold and use a lower value 0.025, a minimum of 7% anomaly mixture-rate yields 100% detection rate as shown in Table 3.2. This suggests that the threshold can be manually tuned based on practical KLD values to improve the detection on low-rate anomalies.

Table 3.1: False Negative Rate with 0.05 as Threshold

Mixture-rate	**False Negative Rate**	**Average KLD**
0.06	1.0	0.0231
0.08	1.0	0.0337
0.10	0.93	0.0468
0.12	**0**	**0.0599**
0.14	0	0.0738
0.16	0	0.0887

Table 3.2: False Negative Rate with 0.025 as Threshold

Mixture-rate	**False Negative Rate**	**Average KLD**
0.05	1.0	0.0337
0.06	0.7143	0.0239
0.07	**0**	**0.0278**
0.08	0	0.0335
0.09	0	0.0407
0.10	0	0.0474

3.3 Summary

In this chapter, GMM and KLD methods are applied and can provide numerical analysis on the change of self-organizing maps. However, we have to admit that SOM-based algorithms have limitation on the scalability of the training dataset. As the number of flows in the training dataset become larger, it is hard to show all the flows as separated clustering regions on a single map. Due to the similarity between the attack flows and the normal flows, it is possible that the anomalous inputs are mapped to the same regions as the normal inputs. In this case, the mixed inputs need to be retrained on a new layer of SOM. Therefore, suitable hierarchical and distributed SOM algorithms need to be developed to solve the problem. We will discuss the details of hierarchical SOM in Chapter 5.

Chapter 4

Hierarchical and Distributed SOM

Self-Organizing Maps can be used for anomaly detection with unsupervised training. Basically, before the training it is not necessary to know the clustering in the trained dataset, because with the output hit maps, we can learn the inner correlations among the trained inputs and visualize the clusters as shown in Fig. 4.1. Once the training phase is finished, we can see how many clusters and the number of hits/inputs are there in each cluster. Therefore, if we suppose that the training dataset contains the normal traffic only, we can use the trained self-organizing maps to detect future anomalies by computing the Best Matching Units (BMUs) for new inputs. As shown in Fig. 4.1, if the BMUs for the new inputs are not located in any of the regions on the trained hit maps, then it will be regarded as anomaly and trigger the alarm for the anomaly detection.

However, even if the BMUs of the new inputs can be located in the regions on the trained map, we cannot guarantee that the inputs belongs to the normal traffic. This is due to the similarity of the traffic which the attackers try to imitate and the native statistical variances from the inputs. In order to improve the accuracy of the anomaly detection, we need to zoom in some of the "mixed" regions, i.e., a subset of the original inputs need to be extracted and trained for new layer(s) of self-organizing maps. This introduces the hierarchical design of SOM and we will discuss our proposed solution in this chapter.

4.0.1 Previous Work on Hierarchical SOMs

There are studies in the literature that discuss the hierarchical SOM algorithm. One popular is named as "Growing Hierarchical Self-Organizing Map" (GHSOM) [5] pro-

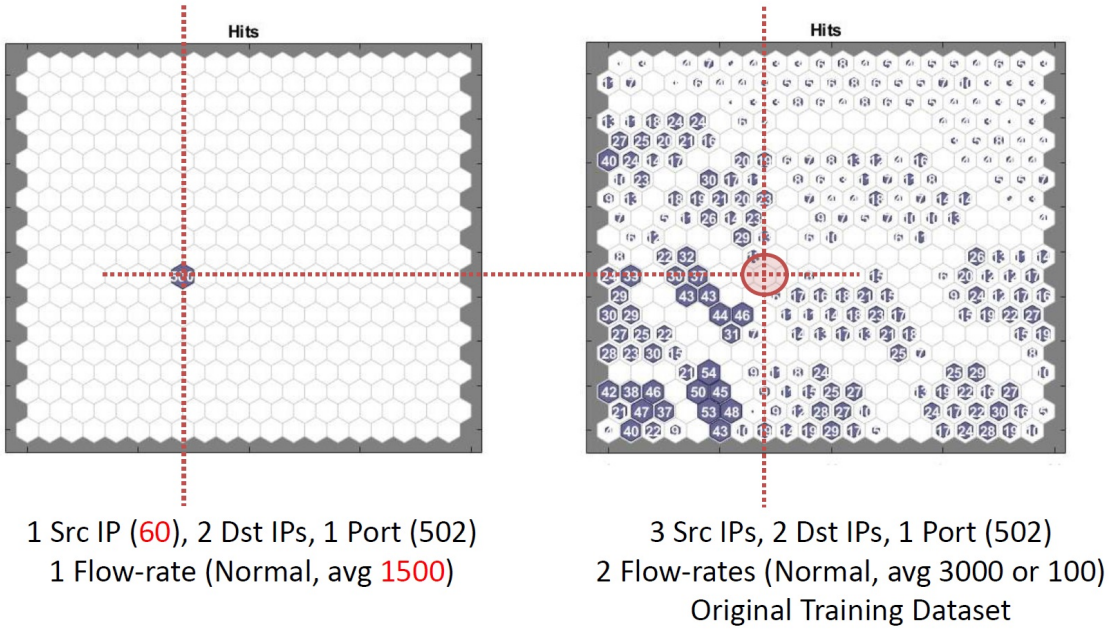


Figure 4.1: Pin-point the Anomaly on the Hit Maps

viding a problem-dependent model that can adapt its architecture during its unsupervised training process according to the particular requirements of the input data. As shown in Fig. 4.2, the map in layer 1 provides a rather rough organization of the main clusters in the input data while the six independent maps in the second layer offer a more detailed view of the data. The third-layer maps are expanded from the two units in one of the second-layer maps and so on. It was shown in [5] that with GHSOM, the overall training time is largely reduced since only the necessary number of units are developed to organize the data collection at a certain degree of detail.

Other studies such as [41] and [42] also discuss similar hierarchical architectures to apply the SOM algorithm. In [41] the algorithm was applied to the detection of novel human behavior in indoor environments. Their system can unsupervisedly learn normal activity and then report novel behavioral patterns as abnormal. In [42], their algorithm was applied to the network intrusion detection based on the KDD dataset [43]. Based on all 41-features from the KDD dataset, they demonstrated that best performance can be achieved using a two-layer SOM hierarchy. It was also shown in [42] that in the training of the second layer of SOM hierarchy, a subset of certain inputs were extracted so that the new training could separate the anomaly inputs from the normal inputs. However, based on our experiments in this chapter, there is no significant help on the separation unless we add labels of normal and abnormal as

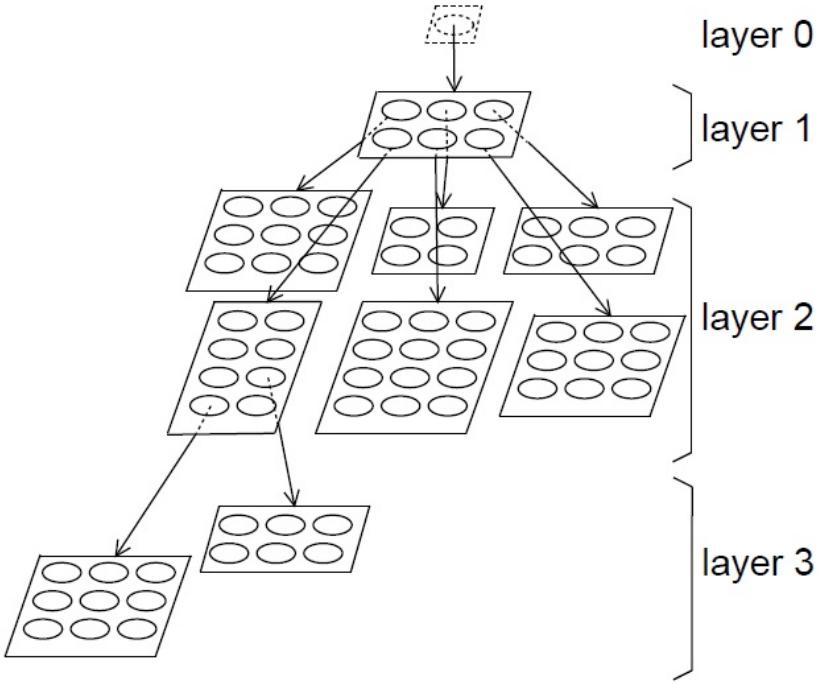


Figure 4.2: Hierarchical structure of GHSOM [5]

a new variable to the extracted inputs. We will discuss this problem in more details in Section 4.1.

We also suggest to use the Principle Component Analysis (PCA) before separating the features, as correlations between different features may exist. For example, one obvious fact is that the number of packets per second will increase with the rise of the number of bytes per second, and these two traffic features should not be treated as independent. Moreover, there may exist hidden correlations among the traffic features such as the relationship between the number of simultaneous connections and the number of IP addresses and port numbers in the network. Without the analysis on the independence among the traffic features, we cannot simply separate them for different SOM training. However, with the PCA on the multiple traffic features we can reduce the dimension of the training dataset and conserve the inner-correlations at the same time.

4.1 Anomaly Detection with Hierarchical SOM

In this section, we will show how to generate new layer(s) based on the first initial SOM training in our approach. The regions or cells that absorb both the anomaly and normal inputs on the map need to be selected as the training inputs for the new layer. We will also justify that with a new dimension being added in the second layer training, it will be clearly enough to separate the anomalous and normal traffic on the new layer. The new dimension is suggested to be chosen based on the labels of the inputs. Due to the similarity between the anomaly and the normal inputs, simple zoom in operation and training again with the inputs in the mixed region cannot work well unless the labels are added.

Before building a multi-layer SOM training algorithm, it is also interesting to see the hierarchies in SOM. The cells from the output maps are inner correlated (e.g. the neighborhood relations) . Some of the cells are close in distance and this is how the clusters are formed. Once a neural cell absorbs both the anomalous and normal inputs which turns out to be a "mixed cell", it is highly possible that the neighbor cells in the same cluster becomes a "mixed cell" as well.

4.1.1 Hierarchies in SOM

With the SOM training, we can see the number of clusters in the training dataset with visual inspection. However, we can answer the same question with a more precise approach. We can use the SOM toolbox [34] in Matlab to show the relationships among the cells on the map with a hierarchical tree structure. This process will first be explained by going through an example in which 3 main clusters can be detected.

Figure 4.3 shows the output after the SOM training. The original dataset has 3 dimensions (coordinates x, y and z) and can be classified in 3 clusters as we can see in Fig. 4.3. There are 3000 samples/observations in the training dataset, and natively we can use the "linkage" function in Matlab software to build a hierarchical tree with the raw data. As shown in Fig. 4.4, the hierarchical tree is built from the leaves, each of which is the single observation from the total 3000 samples. The leaf nodes are treated as the base clusters, i.e. 3000 single clusters on the bottom from which the higher layer clusters are built. The x axis stands for the number of samples and the y axis stands for the Euclidean distance between two clusters. Two leaf nodes are grouped together if the Euclidean distance is small. We can see there are 3 main

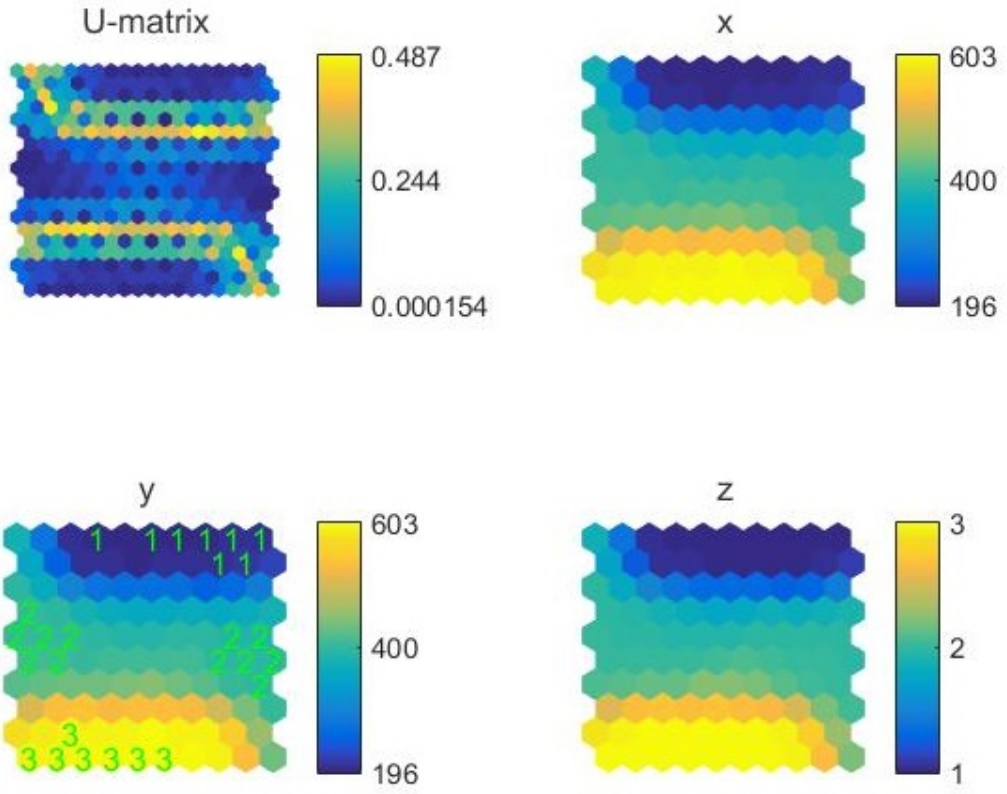


Figure 4.3: SOM Training for 3 clusters

clusters on the second top level, with large distances between each of the clusters as shown in Fig. 4.4.

However, creating a hierarchical tree from the raw data is time consuming. In our above experiment, it is on the order of several minutes (usually less than 5 minutes with our Intel i7 2600K 3.4GHz CPU) to generate the tree with 3000 samples. If the size of the training data becomes larger, the building time will increase because the number of leaves is equal to the number of the samples. Also, the tree structure can become more sophisticated to read with more nodes. Therefore, we suggest to build the hierarchical tree on top of the SOM training. No matter how large the dataset is, we can always build the tree structure with the neural cells from the output maps, and the number of neural cells is usually small (e.g. 100 cells from a 10 by 10 map). Figure 4.5 shows the tree structure built from the maps after SOM training and we can find the same result with 3 main clusters detected. This also shows that the SOM training is self-adaptive and make the organization of the cells on the map to reveal the cluster patterns of the training dataset.

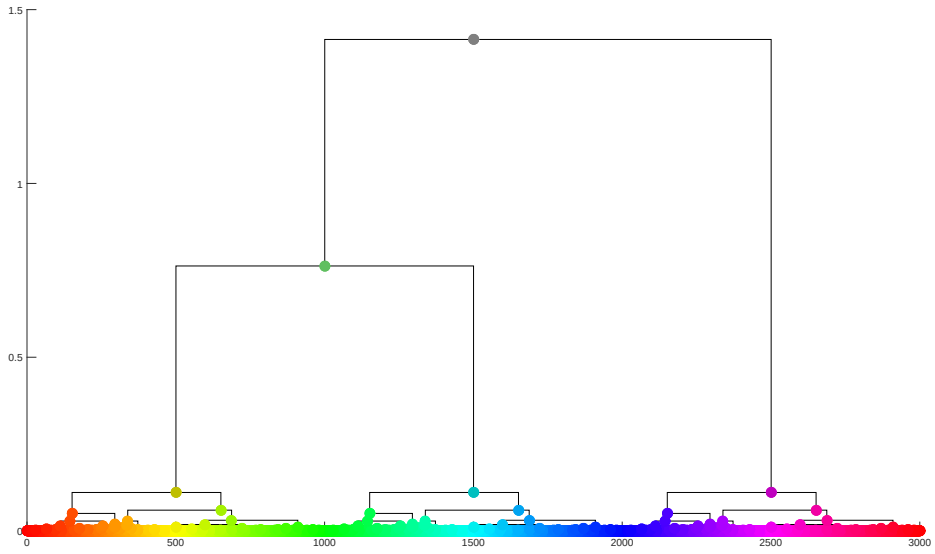


Figure 4.4: Hierarchies in Raw Dataset (3 main clusters)

With the hierarchical tree structure built on top of the SOM training, we can also see the inner correlations for each pair of the cells. Some pairs have small distance which is less than 0.025. However, if we set a threshold of 0.1, there will be 4 clusters as shown in Fig. 4.5. This means that the understanding of the clustering on the map depends on the numerical threshold on the distance. Suitable threshold (e.g. 0.15) can be chosen manually but it can be difficult when the scale of the dataset becomes larger. Nevertheless, it is still useful to see the hierarchies in the SOM because we can mark the cells with different labels according to the tree structure. In the future training, if the anomalous inputs are mapped in the cells which already have the normal inputs, we can pinpoint the locations of the attacked cells on the tree. If the attacked cells have large number of neighbor cells (the ones with small distance), then the attack can be labeled as high level risk, otherwise if the attacked cells are apart from the other cells, then the attack is in low level risk.

4.1.2 Building A Two-layer SOM Architecture

In order to separate both the anomalous and normal inputs are mapped to the same cells with the original SOM training, we need to know what are the mixed inputs and

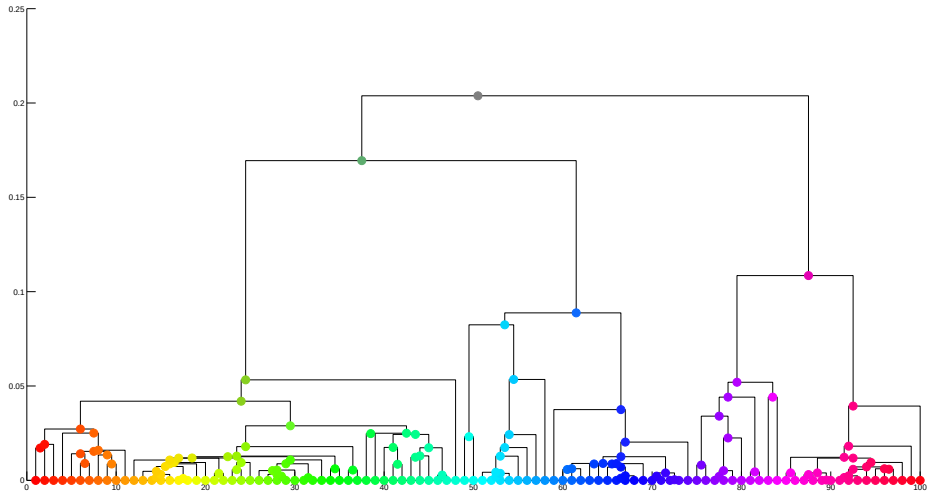


Figure 4.5: Hierarchies in SOM (3 main clusters)

build a new layer of SOM with the subset of these mixed inputs. Figure 4.6 shows the process of creating the second layer of SOM.

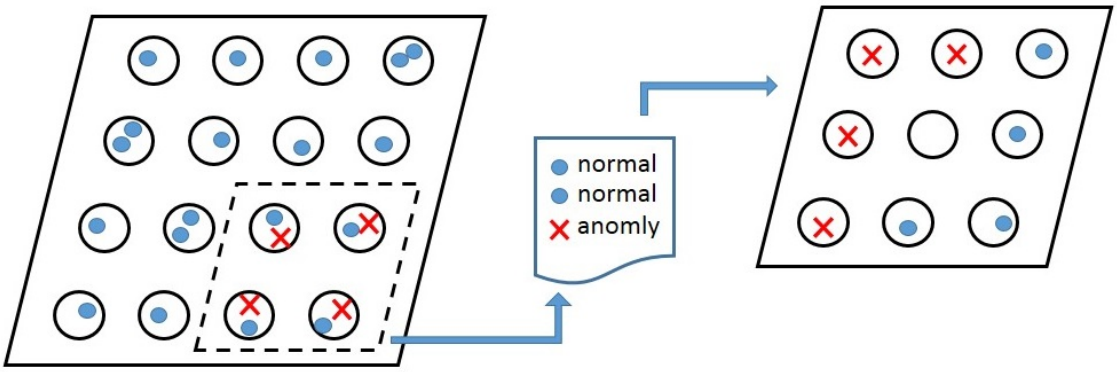


Figure 4.6: Two-layer SOM Training

As discussed in [42], the second layer training can help to separate the mixed inputs. However, we conducted a few experiments with the real CTU-13 trace data [40], and with only the mixed inputs there is no significant boundary to appear in the second layer training. As shown in Fig. 4.7, the anomalous inputs are mapped to the cells which are linked to the normal inputs. After we extract the mixed inputs, i.e., both the anomalous and normal inputs in the "mixed cell", a second layer SOM

is trained with this subset of inputs. However, as shown in Fig. 4.8, even though we can see some of the anomalous inputs are separately mapped to different regions on the second layer map, the majority of the anomalous inputs are still mixed with the normal inputs. This is due to the similarity between the anomalous and normal inputs, otherwise it will be clear to separate them in the original SOM training. Thus we suggest to add new labels to the mixed inputs in order to help the separation.

4.1.3 Training with Labeled Inputs

In our experiments, the labels are only the tags of "attack" or "normal" and are translated as integers of "0" or "1". This means that the dimension of the mixed inputs will be increased by 1. However, the new dimension will significantly help to improve the accuracy of anomaly detection by clearly separating the anomalous and normal inputs. Figure 4.9 shows the effectiveness of adding the labels. The anomalous and normal inputs in the new training are separated clearly and thus in the future it can provide a higher accuracy to detect the anomaly.

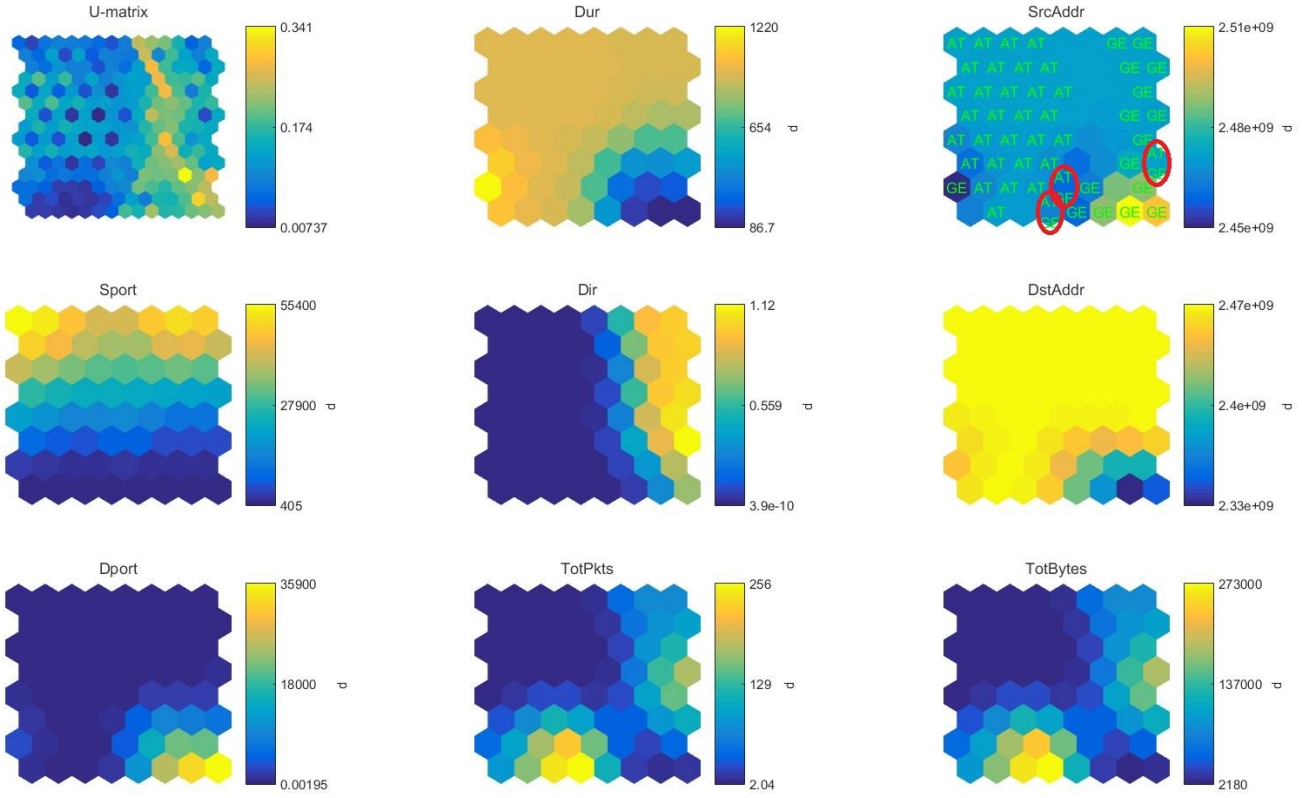


Figure 4.7: Mixed Inputs on the Map (Original Training)

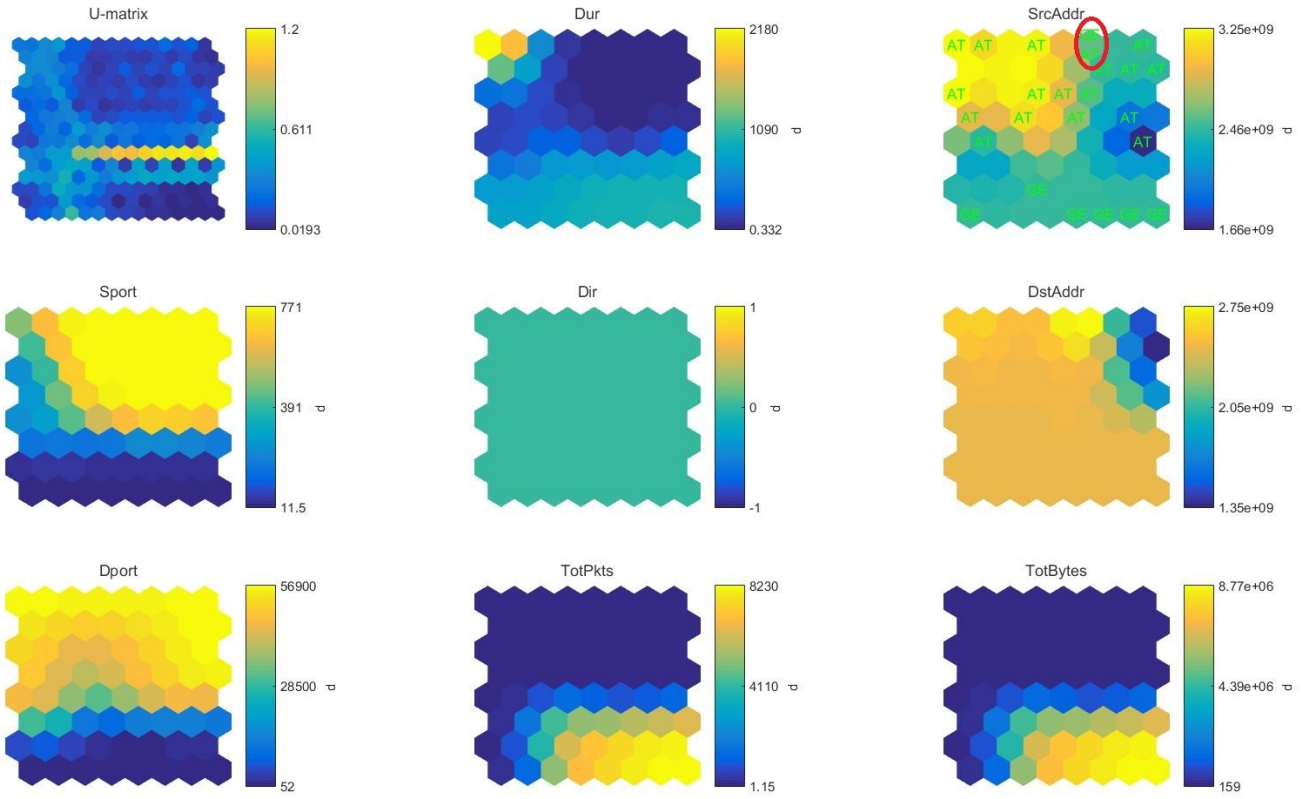


Figure 4.8: Mixed Inputs on the Map (Second Training)

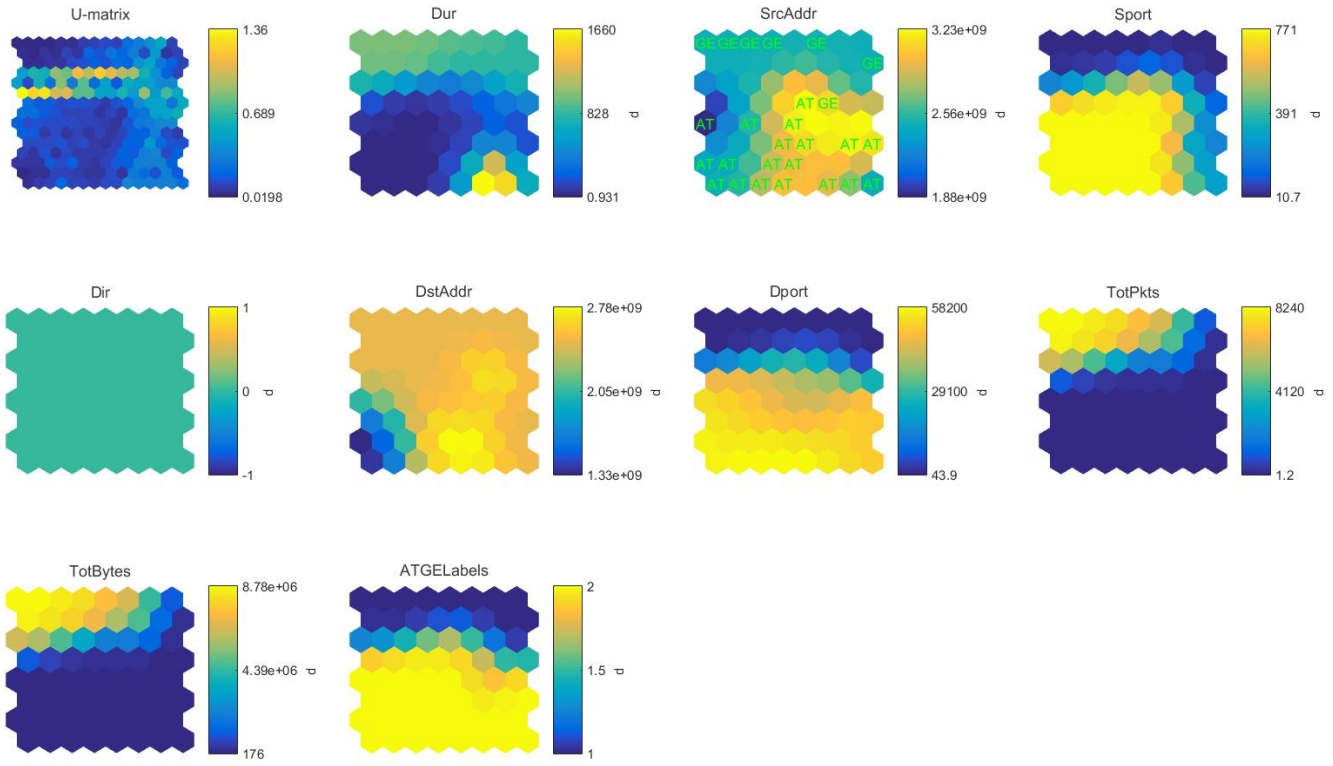


Figure 4.9: Separate Inputs on the Map (Second Training with Labels)

4.2 Dependency Conserving Preprocessing Approach

In this section, we will discuss how to conserve the dependency while the inputs are trained with SOM. As discussed in [42], their design is to train SOM with individual vectors from the bottom layer. This can provide a distributed scheme for the SOM training and reduce the training time. However, it will also break the inner correlations among different feature vectors (e.g. the number of simultaneous connections and the number of IP addresses). Some of the features might even have non-linear or inconspicuous correlations which makes the case even worse if these features are separated and trained individually.

Separating the variables without dependency checking could also add flaws to the anomaly detection system. For example, we can suppose there are two flows exist in the network. One flow has a high flow-rate but it only uses few number of ports, while the one with a low flow-rate uses large number of ports. If we treat the network feature flow-rate independently with the number of ports being used, then it is hard to detect the anomaly which has a high flow-rate. Because in this case, if the attacker tries to imitate the one with higher flow-rate but with exhaustion of port resources, neither the flow-rate nor the number of ports being used on the separated SOM training shows the variations. In order to conserve the dependency among different features, we suggest to apply the Principal Component Analysis (PCA) for pre-processing.

4.2.1 Principal Component Analysis

Principal Component Analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called "principal components" [44]. PCA is frequently used in various real world problems and the applications of PCA have been in representing the data using smaller number of variables [45]. The orthogonal linear transformation will transform the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate and so on. PCA computes the most meaningful *basis* to express the data. As a basis is formed by a set of linearly independent vectors, PCA returns a new basis which is a linear combination of the original basis.

PCA is the simplest of the true Eigenvector-based multivariate analysis and can be done by eigenvalue decomposition of a data covariance matrix or singular value

decomposition of a data matrix, usually after mean centering and normalizing the data matrix for each attribute [46]. The procedure of PCA is listed as follows:

1. Calculate the sample covariance matrix A for the original dataset $\{X_{i,j}\}$, where $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$.

2. Calculate the eigenvectors $\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_n}$ and eigenvalues $\lambda_\mathbf{1}, \lambda_\mathbf{2}, ..., \lambda_\mathbf{n}$ of the covariance matrix.

3. Choose the eigenvectors corresponding to the p largest eigenvalues. The sum of the proportion of variance of the chosen vectors should usually reach more than 80%, where the proportion of variance V_k for the k-th top eigenvalue is computed as:
$$V_k = \frac{\lambda_k}{\sum_{j=1}^{n} \lambda_j}.$$

4. Standardize the original variables by subtracting its mean from that variable and dividing it by its standard deviation:
$$Z = \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_p \end{pmatrix},$$
$$Z_i = \{Z_{i,1}, Z_{i,2}, \cdots, Z_{i,n}\},$$
and
$$Z_{ij} = \frac{X_{ij} - \bar{x_j}}{s_j},$$
$$x_j = \frac{\sum_{i=1}^{m} X_{ij}}{m},$$
$$s_j = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_{ij} - \bar{x_j})^2},$$
where X_{ij} is the data for variable j in sample unit i, $\bar{x_j}$ is the sample mean for variable j, and s_j is the sample standard deviation for variable j.

5. The transformed variables $\hat{Y}$ are calculated as:
$$Y_1 = e_{11}Z_1 + e_{12}Z_2 + ... + e_{1p}Z_p$$

$$Y_2 = e_{21}Z_1 + e_{22}Z_2 + ... + e_{2p}Z_p$$

$$\vdots$$

$$Y_p = e_{p1}Z_1 + e_{p2}Z_2 + ... + e_{pp}Z_p$$

4.2.2 Dependency Conservation on SOM

As we suggest to apply PCA before SOM training, the transformed dataset can be used as the inputs for SOM. This is justified through the following experiments. Figure 4.10 shows the SOM training with the original dataset which has 8 features/variables in total. Applying the PCA on the original dataset, we can obtain the eigenvalues and the proportion of variance for each eigenvalue. As shown in Fig. 4.11, the cumulative proportion for the first 3 eigenvalue out of 8 already reaches 85.92%. Therefore, we choose the first three principal components to represent the original dataset, i.e., the number of variables in transformed dataset has been reduced from 8 to 3.

The transformed dataset can be later used as the inputs for SOM training, and Fig. 4.12 shows the training results with the top 3 principal components. However, as the number of variables reduced from 8 to 3, it is helpful to see the difference between the training with transformed dataset and the training with the original dataset. Then we utilized the KLD method as discussed in Chapter 2 to show the difference with fewer variables. Figure 4.14 shows the KLD number as 0.22 which is a small number less than 1 as discussed in Chapter 2. This means that with only 3 components that covers 85.92% variance of the original dataset, the transformed dataset can still represent the principal features, while it can significantly reduce the training time as the number of variables or the dimensions of the dataset is reduce from 8 to 3. We further investigate the performance of choosing more principal components. As shown in Fig. 4.13, the dimension of the original dataset is only reduced by 1. With 7 chosen PCs, the original dataset is transformed and trained again with SOM. The KLD number becomes smaller as shown in Fig. 4.15, which is expected because we conserve more principal features from the original dataset. Finally, we also see the case that 8 principal components are all chose from the eigenvector. The SOM training with the transformed dataset is shown in Fig. 4.16, which turns out to be similar as the original training. This is justified by the KLD number 0 as shown in Fig. 4.17.

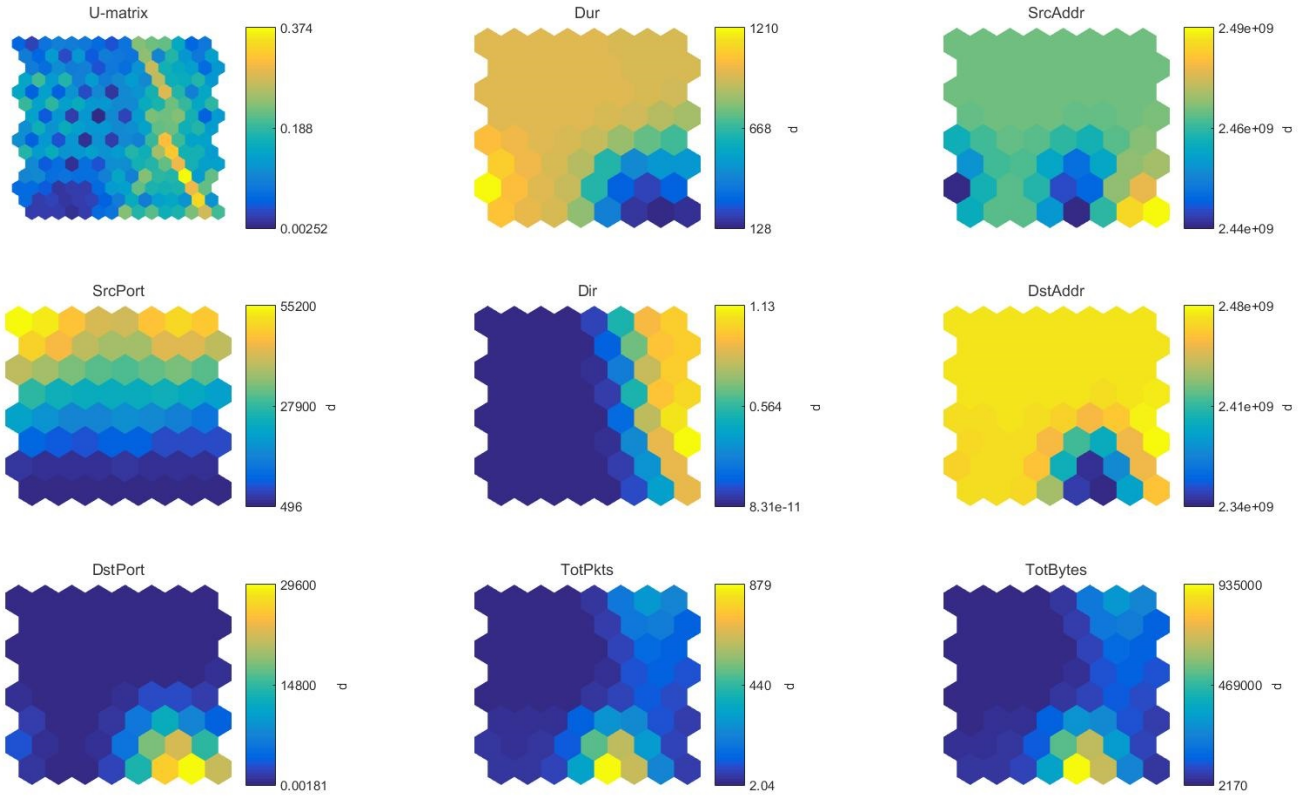


Figure 4.10: SOM Training with 8 Features (CTU-13)

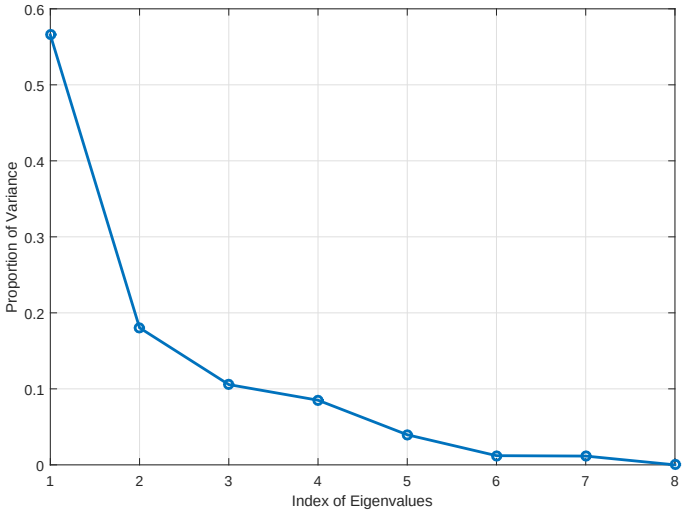


Figure 4.11: Proportion of Variance from Eigenvalues

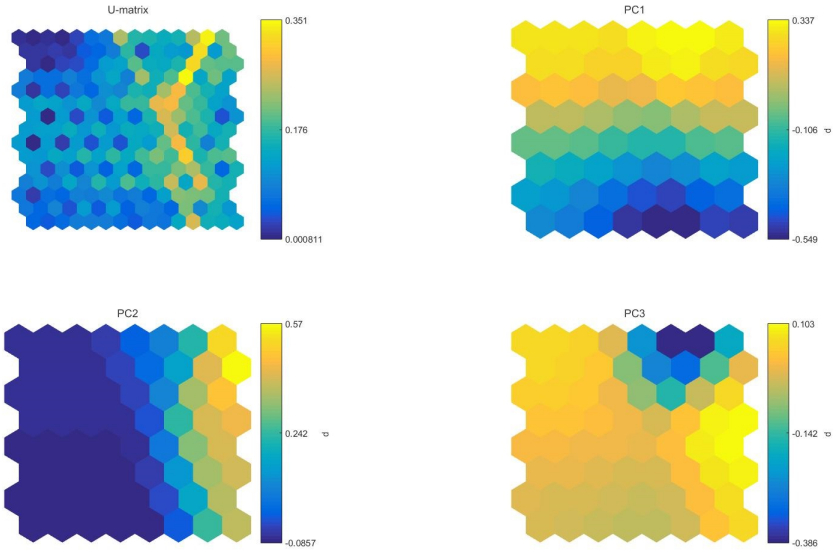


Figure 4.12: SOM Training with 3 PCs (CTU-13)

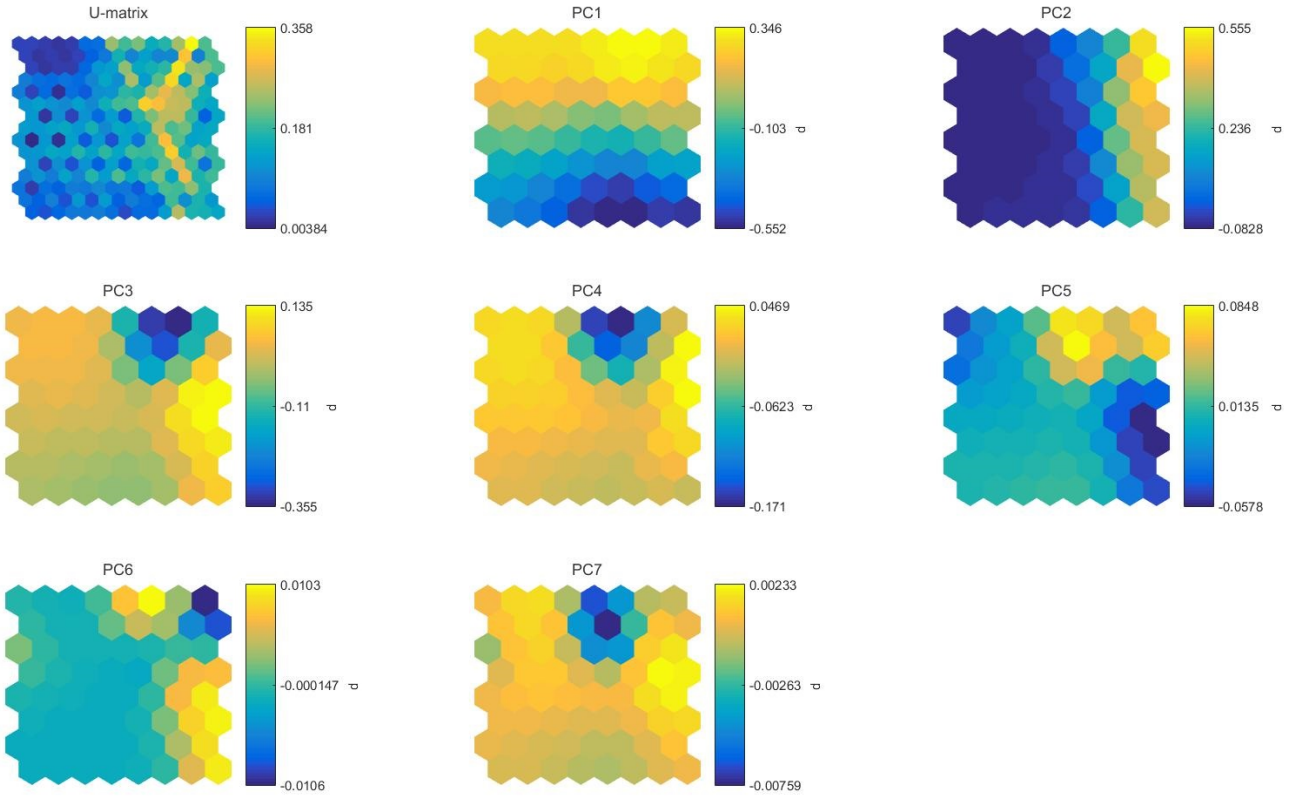


Figure 4.13: SOM Training with 7 PCs (CTU-13)

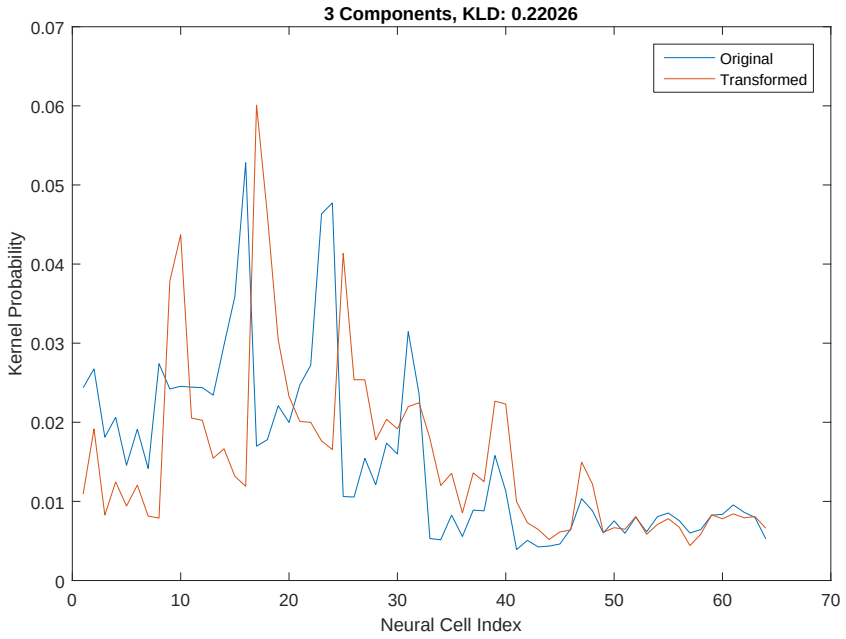


Figure 4.14: KLD with 3 PCs (CTU-13)

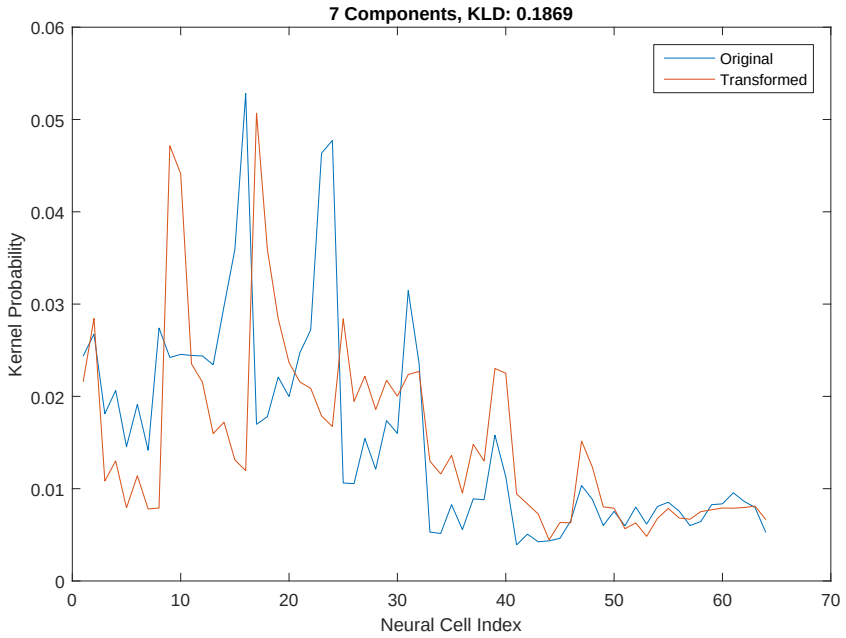


Figure 4.15: KLD with 7 PCs (CTU-13)

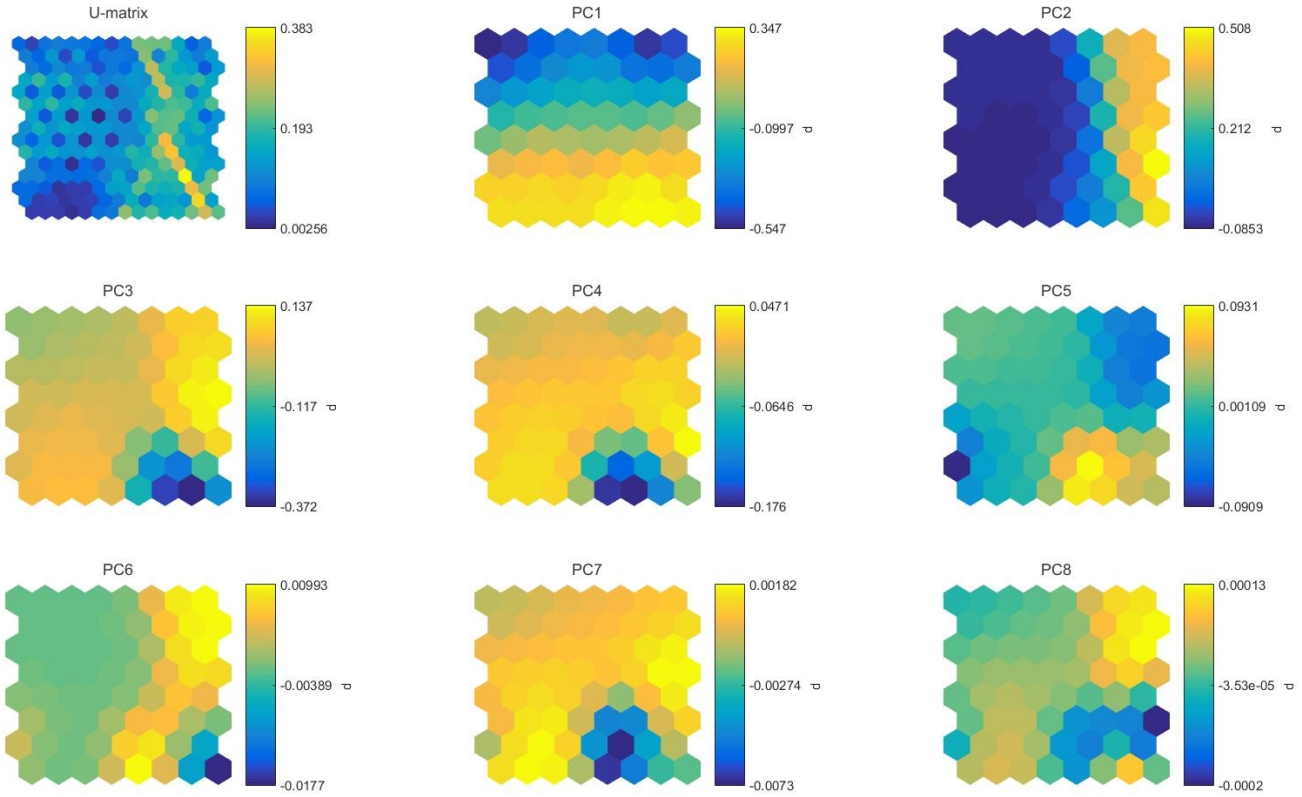


Figure 4.16: SOM Training with 8 PCs (CTU-13)

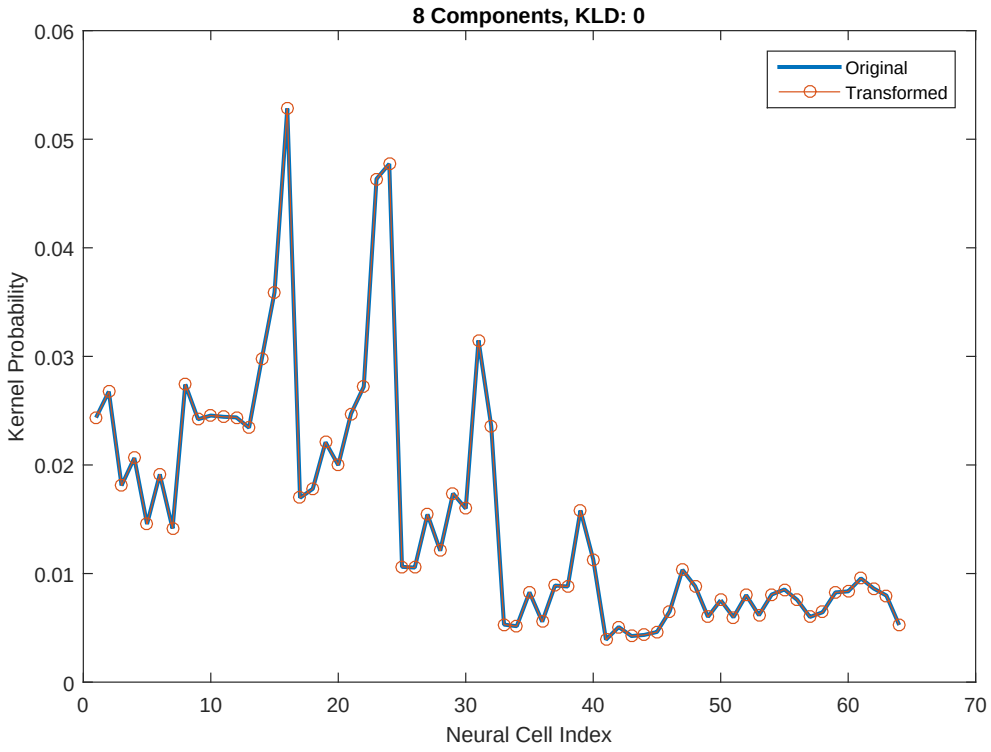


Figure 4.17: KLD with 8 PCs (CTU-13)

4.3 Summary

In this chapter, we further extend the SOM-based approach with a two-layer hierarchical design. In order to increase the accuracy for anomaly detection, the inputs which cause the "mixed-cell" need to be extracted and re-trained with a new layer of self-organizing map. We point out the adding of a new dimension with labels can significantly help the separation for the mixed inputs. We further investigate on how to conserve the inner correlations within the original dataset instead of simply separate the variables for distributed SOM training. To solve this problem, PCA is suggested to be used and justified.The transformed dataset can maintain a manageable level of variances from the dataset, reducing the dimensions from the original training and conserving the dependencies at the same time.

Chapter 5

Enhancing Self-Adapting Caching with SOM in Wireless SCADA Networks with Mobile Clients

This chapter discusses a wireless data acquisition application for SCADA networks. SCADA networks with wireless clients and sensors are the trend for the future as discussed in [47, 48]. However, considering the wireless accesses for the monitoring data and the mobility of the clients or sensors, new approaches of organizing the storage of the SCADA networks should be developed. Taking advantage of a mobility-awareness caching scheme for fast data acquisition in the Information-Centric Networks (ICNs) [49], we further extend that approach with SOM learning in a scenario of the SCADA network with wireless accesses from mobile clients.

5.1 Introduction on Wireless SCADA Networks

SCADA systems are generally implemented for industry control and management based on the information collected from the sensors in the field. The data collected from the sensors is usually sent to a central computer/server for the analysis. Meanwhile, people equipped with the client device can also access to the data for the real-time monitoring. However, as the number of sensors keep increasing with the extension of more controlled areas, it is possible to see a case where wired communication is hard or impractical to be implemented. Thus the integration of wireless communication in SCADA network is introduced in recent years [50]. With the sup-

port of mobile devices and the network integration with the Internet, clients can also access the data from the central server with wireless connections. This is also beneficial especially for real-time monitoring. For example, in a case that people need to check the field areas regularly, with the wireless accessibility they can download the data (e.g. camera vidoes) for the interested areas remotely when they are on the road.

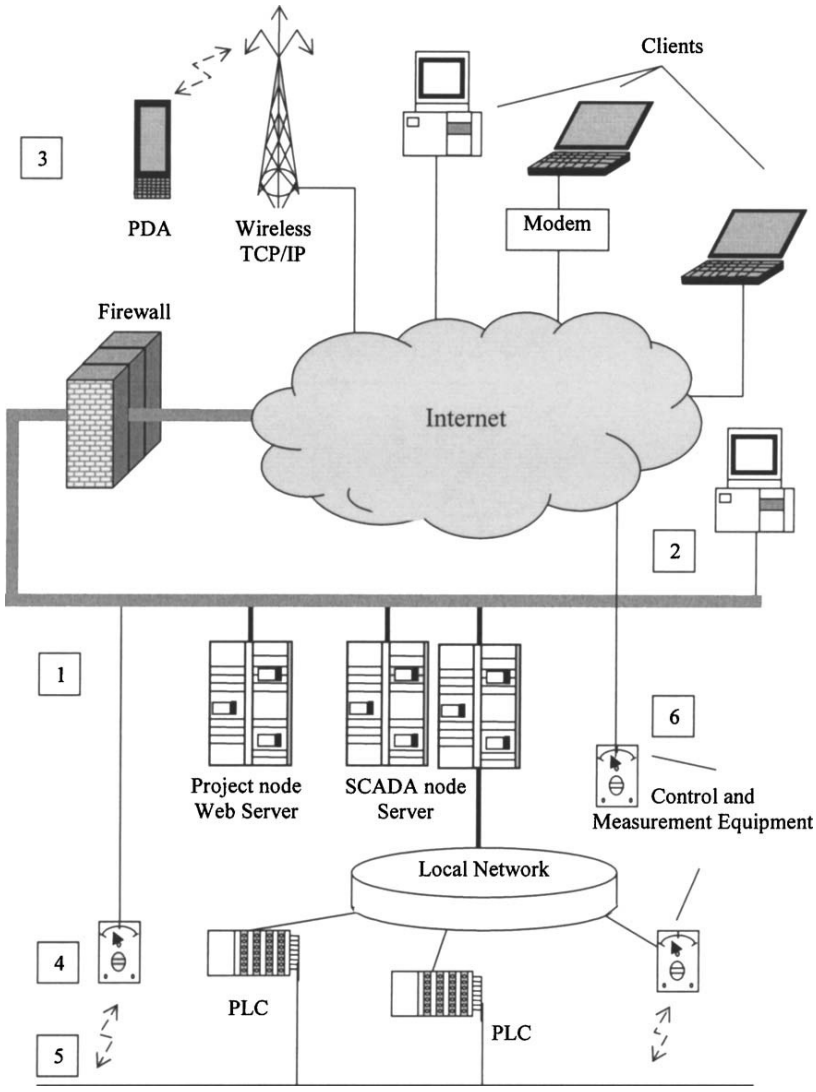


Figure 5.1: Components of a Web access based SCADA system [6]

There has been some study on the architecture of SCADA networks with wireless accessibility. In [50] an architecture of SCADA in wireless mode is described, in which it extends the fieldbus to the Internet and PLCs and RTUs can connect to the SCADA through wireless communication to the Internet Gateway. A mobile phone

based SCADA was also proposed in [6], supporting the online control of the prototype crane via mobile phones or PDAs as shown in Fig. 5.1.

As we introduce new architectures for SCADA networks with the support of wireless communication, new problems such as how to optimally locate the data on different sites of the SCADA networks rather than storing one at the central server, how to support the wireless data requests from mobile clients and how to take advantage of the mobility of the clients in the design of caching schemes in a data-oriented SCADA system need to be addressed. In this chapter, we focus on the problem of dynamic caching for the requests of data from mobile clients. A SOM-based approach is proposed based on a method discussed in [49], where the original method is designed for the ICNs but with similar application scenario. As we focus on the data-oriented service provisioning, we suppose the topology in the original model can be applied to the scenario of wireless SCADA networks, which is data-centric and with mobile clients (e.g. guardians who need to check different sites of a SCADA network regularly) accessing specific parts of data remotely.

On the client side, with the rapid development of the WiFi and 3G/LTE technology, mobile devices such as cell phones, tablets and laptops have become pervasive in people's daily life and have been widely used to connect to the Internet. If the SCADA networks can support the integration of the Internet, mobile clients can access to the SCADA network nearly from anywhere. As described in [49], places such as university campuses, airports and industrial plants can provide WiFi accesses so that mobile users can connect their devices through WiFi. We suppose the same for the scenario of wireless SCADA networks, i.e., there are Access Points (APs) in different areas of the network and the users can access the data stored on the servers in the SCADA networks.

More specifically, if we consider a local-scale activity such as the scheduled guarding or infrastructure checking across different sites of the field, it is more common to see that network users have a tendency to move across different buildings or areas of the network that are covered with WiFi access points. Such a user mobility has introduced new challenges to SCADA data-oriented service provisioning. In [49], the file contents were traditionally cached at distributed ICN servers, following their own popularity and a mobility-awareness caching strategy was designed to increase the cache hit rate. We suppose the SCADA network follow the same strategy for data storage in order to support fast data acquisition, backup and recovery.

However, in order to achieve an optimal caching strategy, [49] uses the solution of a linear programming approach in the cache scheme. It is time consuming to solve the linear problem as the scalability of the content size and site numbers increase. Therefor, in this chapter we build a SOM-based approach for the sub-optimal caching, which has little difference from the optimal caching computed based on the solution of the linear programming problem. The same mathematical models are applied to characterize the impact of user mobility on the content popularity (similar to popular videos from specific secure places in the guardian systems) at different sites of SCADA networks, with homogeneous and heterogeneous content sizes. With the SOM-based training, it can save the computing time and increase the efficiency of the system at the same time.

5.2 SOM-based Caching Strategy

In this section we will introduce our proposed SOM-based caching strategy. As discussed in [49], the Mobility-Awareness Caching (MAC) approach is computationally intensive. During each update of the caching space on different sites of the network, an Integer Linear Programming problem (ILP) needs to be solved. However, if we utilize the SOM algorithm, only one training is needed based on the history of optimal caching strategies. Although the accuracy is lower compared to the computed optimal caching strategy, the proposed SOM-based algorithm can save time and still provide a sub-optimal performance compared to the original MAC approach.

5.2.1 Approximate the Unknowns with Self-Organizing Map

In the previous chapters we have discussed multiple uses of self-organizing maps, such as finding the anomaly from the output map and discovering the inner correlation between different features/vectors in the training datasets with multi-layer training strategy. However, in this chapter we will use the SOM algorithm for prediction of unknown or missing values. This is an important problem as we cannot always guarantee the integrity of the datasets for training. Due to the physical issues that might appear in the SCADA networks, such as the loss of packets during the transition, it is possible that some broken records (e.g. there are some values missing) exist in the captured datasets. In most cases, people have to remove the broken records during the pre-processing. However, instead of removing or wasting these incomplete record-

s, there are some approaches to recover the **missing values** from on the dataset. In the literature, methods such as calculating the average value of the field/vector in which the missing value exists, or doing linear or quadratic regression with the rest of data are really popular. But due to the low accuracy and efficiency of these traditional methods when the data cannot be simply understood with linear or quadratic relationships, new methods need to be developed.

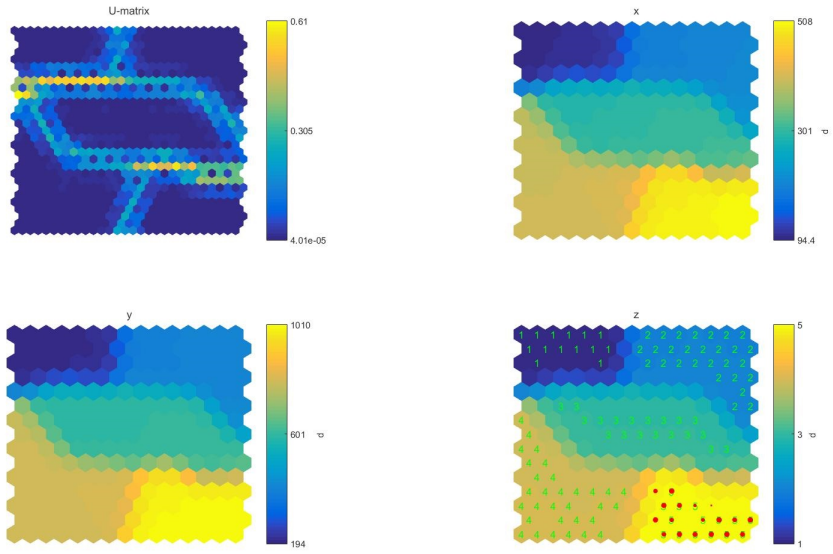


Figure 5.2: Approximate the Unknowns with SOM Algorithm (Missing the Labels for the 5th Cluster)

SOM-based algorithm can natively support the multi-dimensions of the training dataset. As discussed in [51], the SOM model can reduce the time of clustering and do well at the same time for estimating the missing values with the help other inexpensive variables. Also the study of [52] shows that using the SOM which is unsupervised neural networks to estimate the missing values and replace outliers in time series data is simple, computationally efficient and highly accurate. Due to these promising features of the SOM-based algorithms, we utilize the self-organizing maps for the caching scheme, which means the caching strategy for certain content will be decided based on the SOM training rather than solving the linear programming problem modeled in [49].

Figure 5.2 shows an example of the prediction results with SOM algorithm. We create a small training dataset which contains 3 variables, x, y, z. The values of x

are generated from 5 gaussian distributions with different mean value but the same standard deviation. The value of y is linearly scaled from the value of x for each record and the values of z are integers and only range from 1 to 5 according to the ID number of each gaussian distribution. Therefore, there are 5 clusters in the training dataset. After the SOM training, there are exactly 5 clusters shown in the self-organizing maps which is expected. In the following procedure, we create a new testing dataset which also has 3 variables x, y, z from the original dataset, but with the values of z missing. Then we can utilize the SOM algorithm to find out the Best Matching Unit (BMU) for the records which is lack of the z values as discussed in [51]. This is due to the training of SOM which makes the similar records closer to each other, i.e., we can suppose that the records mapped to the same BMU contains similar values on x, y, z. Once the BMU is found for a specific record with value of z missing, the missing value can be **approximated** based on the values of z in other records that are mapped to the same BMU. As shown in Fig. 5.2, the predicted values of z are displayed on the bottom right training map. For the records with estimated 5 for z, they are displayed as points to show the locations where the records are mapped. Compared to the values in the original dataset, the predicted missing values of z for each records in the test dataset are exactly the same.

5.2.2 SOM-based Caching Scheme

As shown in Sec.5.2.1, SOM can be used for approximating the missing values. Therefore, we further develop the caching strategy with self-organizing maps. The main objective is to decide whether or not to cache certain content based on its properties. In this case, the computed solution for the optimal caching is used as the training dataset. As we can keep track of the caching strategies in the log history based on the size, site location, popularity, each caching record can be modeled as one record with 4 variables: 1) the size of the content (values in KB); 2) the intended site location (values in the range of 1 to 7); 3) the popularity rank at the intended site (values in the range of 1 to 100); 4) whether or not to cache at the intended site (binary values). In the testing dataset, the values in the 4-th variable are missing and need to be approximated with sub-optimal caching strategies.

In our experiments, we use the cache log of 400 optimal caching records for 60 popular contents/videos in the network. Initially the strategies in the caching records are computed based on the linear programming problem modeled in [49]. Then the

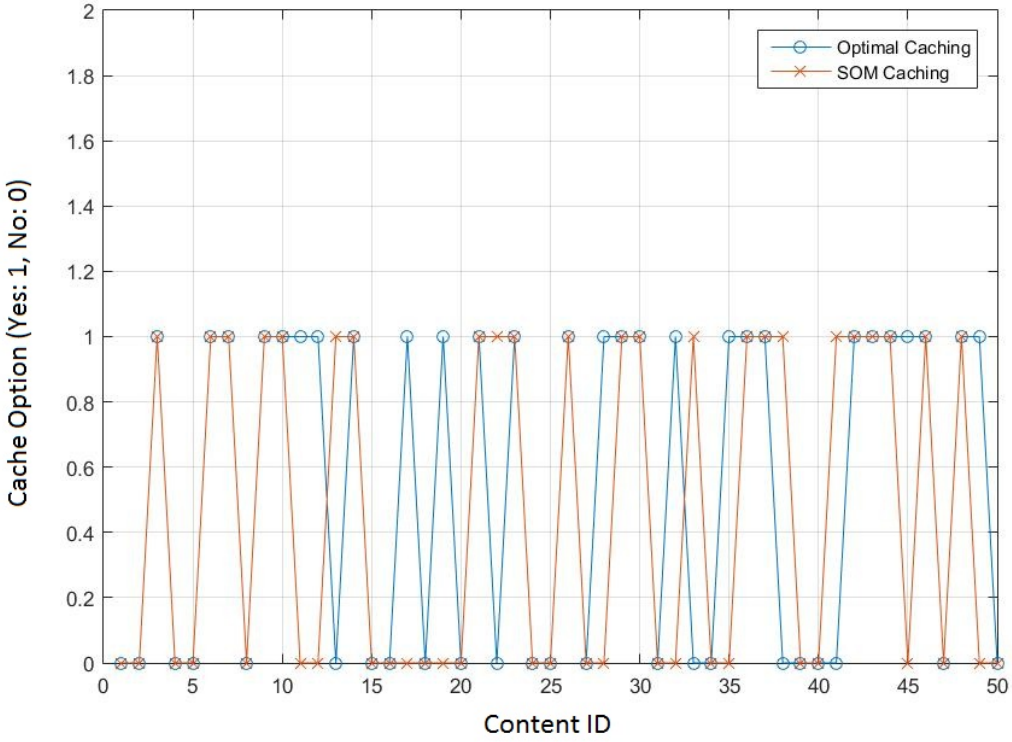


Figure 5.3: Predicting with SOM Algorithm

computed optimal solutions are transformed to the 400 optimal cache records with 4 variables described above. Later the optimal records are used for SOM training which is quick and only takes a few milliseconds. After the SOM training, we randomly select 50 records from the training dataset for testing, however with the values of the 4-th variable, whether or not to cache at intended site removed. With the SOM algorithm, we compute the Best Matching Units (BMUs) for the 50 records and use the values from BMUs to approximate the missing value. Figure 5.3 shows the result of SOM prediction, comparing the "optimal values" which are computed based on the solution of the optimization problem in [49] and the "predict values" which are predicted using SOM. In our test case, the approximation error equals to 28%.

5.3 Summary

In this chapter, we further discuss the use of SOM in the scenario of wireless SCADA network for fast data acquisition. The self-organizing maps can be used as a new tool

for optimizing the content locations. Also, due to the change and update of network, the previous optimization problem needs to be solved multiple times while there is only one training needed with SOM. The accuracy of predicting the locations of cached content can be improved with unsupervised learning on a larger dataset. We believe that the SOM-based algorithms have a promising future for different applications with more extensions and development.

Chapter 6

Conclusions and Future Work

Unlike traditional signature-based approaches, a SOM-based approach can be applied to detect unknown attacks. We have shown that, within an acceptable training time, this provides a new way to produce two dimensional mapping for visual inspection. It is, however, sometimes difficult for humans to perceive the subtle differences that exists between SOM trained maps, which is subject to both the shape of the clusters and the weight values in each neural cell. To assist, we further introduce the Gaussian Mixture Model (GMM) and Kullback-Leibler Divergence (KLD) on top of SOM trained maps. Using the suggested average KLD number as the threshold, our approach uncovers anomalies with an accuracy of 100% in our case study, up to a minimum of 12% anomaly mixture-rate from the CTU-13 dataset. With further tuning, we show how to obtain rates as low as 7%. As the scale of the traces become larger, it becomes more difficult to pinpoint the low-rate anomaly with the same accuracy. There are multiple ways to solve this issue, such as implementing a hierarchical SOM training algorithm [33] or running the SOM algorithm in a distributed way [53]. The design of a suitable hierarchical or distributed SOM is a topic of our future work.

6.1 Contributions

At the end of this thesis, we can answer the 4 questions in 1.3 and our contributions can be summarized as follows:

1. We explored that Self-Organizing Maps (SOM) is a useful tool to provide visualizations. As a case study of anomaly detection in SCADA networks, SOM can be useful for visual inspections with fast unsupervised learning on real traces.

2. An enhanced and quantitative measurement on the variations of SOMs is investigated by applying the Gaussian Mixture Model (GMM) and Kullback-Leibler Divergence (KLD). The main contribution in this dissertation is the addition of numerical methods to enhance change detection in SOM algorithms. We use anomaly detection as an example domain.

3. Further development with hierarchical and distributed SOMs is also discussed in order to provide a more accurate traffic control and analysis with large amounts of data. We also use the SOM algorithm in a 'supervised' way with unsupervised training which can improve the accuracy of detecting known attacks and conserve the capability of finding unknown attacks at the same time.

4. We investigated that SOM algorithm can be applied to other applications. As an extended case study, SOM can provide enhanced self-adapting caching strategies by learning about the history of optimal caching strategies in wireless SCADA networks with mobile clients. SOM can help to solve the time-consuming problem of computing the linear programming problem in the optimized caching.

6.2 Future Work

Self-organizing maps can be applied to many problems that introduce data clustering with unsupervised learning. However, there are limitations on the scalability of applying SOM-based approaches. It can be hard to use SOM when there are many variables in the training dataset (this is real in the data centers of large companies like Oracle and Microsoft) as the size of the maps needs to be set larger and training time would increase proportionally. However, SOM works well when the dataset has a lower dimensionality (tens of variables for example). This is particularly helpful for small controlled networks or sub-networks.

As we discussed in Chapter 3, hierarchies can be introduced in the SOM training and can significantly increase the accuracy of anomaly detection. Also when there are large numbers of variables existing in the raw dataset, separating the variables or reducing the dimension in the training dataset would be a way to solve the scalability problem. PCA is a useful method to reduce the dimension of the dataset and is manageable to choose how much correlations and variances are conserved. As it is linear-independent among different principal components, the separation on the dimension can be done according to the PCs and not on the raw variables. However,

the principal components in the pure PCA are expressed as a linear combination of the variables in the raw dataset and it is possible to lose hiden correlations. New methods such as “Kernel PCA” is designed for Nonlinear Component Analysis [54] and can be further explored.

Bibliography

[1] T. Hegazy and M. Hefeeda, "Industrial automation as a cloud service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, pp. 2750–2763, Oct 2015.

[2] ICSLab, "Capture files from 4sics geek lounge," *Industrial cyber security conference 4SICS*, 2015.

[3] E. Pleijsier, "Towards anomaly detection in scada networks using connection patterns," in *The 18th Twente Student Conference on IT*, pp. 1–6, 2013.

[4] K. Mathioudakis, N. Frangiadakis, A. Merentitis, and V. Gazis, "Towards generic scada simulators: A survey of existing multi-purpose co-simulation platforms, best practices and use-cases," in *Scientific Cooperations International Workshops in Electrical-Electronics Engineering*, pp. 33–39, 2013.

[5] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data," *IEEE Transactions on Neural Networks*, vol. 13, pp. 1331–1341, 2002.

[6] E. Ozdemir and M. Karacor, "Mobile phone based scada for industrial automation," *ISA Transactions*, vol. 45, no. 1, pp. 67 – 75, 2006.

[7] H. Ritter, T. Martinetz, K. Schulten, D. Barsky, M. Tesch, and R. Kates, *Neural computation and self-organizing maps: an introduction*. Addison-Wesley Reading, MA, 1992.

[8] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "On the capability of an som based intrusion detection system," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, pp. 1808–1813, July 2003.

[9] R. R. R. Barbosa, R. Sadre, and A. Pras, "Towards periodicity based anomaly detection in scada networks," in *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pp. 1–4, Sept 2012.

[10] I. Garitano, R. Uribeetxeberria, and U. Zurutuza, *A Review of SCADA Anomaly Detection Systems*, pp. 357–366. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[11] N. Goldenberg and A. Wool, "Accurate modeling of modbus/tcp for intrusion detection in {SCADA} systems," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63 – 75, 2013.

[12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, pp. 15:1–15:58, July 2009.

[13] A. Patcha and J. M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, pp. 3448–3470, Aug. 2007.

[14] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo, "Anomaly detection methods in wired networks: a survey and taxonomy," *Computer Communications*, vol. 27, no. 16, pp. 1569 – 1584, 2004.

[15] N. Duffield, P. Haffner, E. Krishnamurthy, and H. Ringberg, "Rule-based anomaly detection on ip flows," in *Proceedings of INFOCOM*, 2009.

[16] E. M. Roche, "Critical foundations: Protecting america's infrastructures," *Journal of Global Information Technology Management*, vol. 1, no. 1, pp. 49–50, 1998.

[17] J. Bigham, D. Gamez, and N. Lu, *Safeguarding SCADA Systems with Anomaly Detection*, pp. 171–182. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[18] V. M. Igure, S. A. Laughter, and R. D. Williams, "Security issues in scada networks," *Computer Security*, vol. 25, pp. 498–506, Oct. 2006.

[19] A. Patel, Q. Qassim, and C. Wills, "A survey of intrusion detection and prevention systems," *Information Management and Computer Security*, vol. 18, no. 4, pp. 277–290, 2010.

[20] B. Zhu and S. Sastry, "Scada-specific intrusion detection/prevention systems: a survey and taxonomy," in *Proceeding of the 1st Workshop on Secure Control Systems (SCS)*, 2010.

[21] A. N. Mahmood, J. Hu, Z. Tari, and C. Leckie, "Critical infrastructure protection: Resource efficient sampling to improve detection of less frequent patterns in network traffic," *J. Netw. Comput. Appl.*, vol. 33, pp. 491–502, Jul 2010.

[22] M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration*, LISA '99, (Berkeley, CA, USA), pp. 229–238, USENIX Association, 1999.

[23] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of pca for traffic anomaly detection," in *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '07, (New York, NY, USA), pp. 109–120, ACM, 2007.

[24] A. Lakhina, M. Crovella, and C. Diot, "Characterization of network-wide anomalies in traffic flows," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, (New York, NY, USA), pp. 201–206, ACM, 2004.

[25] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang, "An empirical evaluation of entropy-based traffic anomaly detection," in *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, IMC '08, (New York, NY, USA), pp. 151–156, ACM, 2008.

[26] B. C. Rhodes, A. M. James, and D. C. James, "Multiple self-organizing maps for intrusion detection," in *Proceedings of the 23rd national information systems security conference*, pp. 16–19, 2000.

[27] G. C. Tjhai, S. M. Furnell, M. Papadaki, and N. L. Clarke, "A preliminary two-stage alarm correlation and filtering system using som neural network and k-means algorithm," *Computers and Security*, vol. 29, pp. 712–723, Sept. 2010.

[28] P. Stavroulakis and M. Stamp, *Handbook of information and communication security*. Springer Publishing Company, Incorporated, 1st ed., 2010.

[29] A. Lemay, "Defending the scada network controlling the electrical grid from advanced persistent threats," *Doctoral dissertation, University of Montreal*, 2013.

[30] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress, 2006.

[31] A. Varga, "The omnet++ discrete event simulation system," in *Proceedings of the European simulation multiconference (ESM2001)*, vol. 9, p. 65, 2001.

[32] C. Queiroz, A. Mahmood, and Z. Tari, "Scadasim: A framework for building scada simulations," *IEEE Transactions on Smart Grid*, vol. 2, pp. 589–597, Dec 2011.

[33] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "A hierarchical som-based intrusion detection system," *Engineering Applications of Artificial Intelligence*, vol. 20, pp. 439–451, Jun 2007.

[34] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "Self-organizing map in matlab: the som toolbox," in *Proceedings of the Matlab DSP conference*, vol. 99, pp. 16–17, 1999.

[35] C. E. Rasmussen, "The infinite gaussian mixture model," in *Proceedings of 1999 Conference on Neural Information Processing Systems (NIPS)*, pp. 554–560, Dec 1999.

[36] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[37] E. Alhoniemi, J. Himberg, and J. Vesanto, "Probabilistic measures for responses of self-organizing map units," in *In Proceedings of the International ICSC Congress on Computational Intelligence Methods and Applications (CIMA99)*, pp. 286–290, ICSC Academic Press, 1999.

[38] I. Nabney, *NETLAB: algorithms for pattern recognition*. Springer Science & Business Media, 2002.

[39] J. R. Hershey and P. A. Olsen, "Approximating the kullback-leibler divergence between gaussian mixture models," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, vol. 4, pp. 317–320, April 2007.

[40] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers and Security*, vol. 45, pp. 100–123, 2014.

[41] G. I. Parisi and S. Wermter, "Hierarchical som-based detection of novel behavior for 3d human tracking," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–8, Aug 2013.

[42] H. Gunes Kayacik, A. Nur Zincir-Heywood, and M. I. Heywood, "A hierarchical som-based intrusion detection system," *Eng. Appl. Artif. Intell.*, vol. 20, pp. 439–451, June 2007.

[43] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, "The uci kdd archive of large data sets for data mining research and experimentation," *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 2, pp. 81–85, 2000.

[44] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[45] M. Chattopadhyay, P. K. DAN, and S. Mazumdar, "Principal component analysis and self-organizing map for visual clustering of machine-part cell formation in cellular manufacturing system," *Systems Research Forum*, vol. 05, no. 01, pp. 25–51, 2011.

[46] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[47] M. Choi, "Wireless communications for scada systems utilizing mobile nodes," *International Journal of Smart Home*, vol. 7, no. 5, pp. 1–8, 2013.

[48] H. Kim, "Security and vulnerability of scada systems over ip-based wireless sensor networks," *International Journal of Distributed Sensor Networks*, 2012.

[49] T. Wei, L. Chang, B. Yu, and J. Pan, "Mpcs: A mobility/popularity-based caching strategy for information-centric networks," in *2014 IEEE Global Communications Conference*, pp. 4629–4634, Dec 2014.

[50] T. H. Kim, "Integration of wireless scada through the internet," *International Journal of Computers and Communications*, vol. 4, no. 4, pp. 75–82, 2010.

[51] L. Zhang, M. Scholz, A. Mustafa, and R. Harrington, "Application of the self-organizing map as a prediction tool for an integrated constructed wetland agroecosystem treating agricultural runoff," *Bioresource Technology*, vol. 100, no. 2, pp. 559 – 565, 2009.

[52] R. Rustum and A. J. Adeloye, "Replacing outliers and missing values from activated sludge data using kohonen self-organizing map," *Journal of Environmental Engineering*, vol. 133, no. 9, pp. 909–916, 2007.

[53] R. Pascual-Marqui, A. Pascual-Montano, K. Kochi, and J. Carazo, "Smoothly distributed fuzzy c-means: a new self-organizing map," *Pattern Recognition*, vol. 34, no. 12, pp. 2395–2402, 2001.

[54] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.