ANOMALY DETECTION FOR IOT DEVICES USING HIERARCHICAL SELF-ORGANIZING MAPS

by

Muhammadjon Toshpulatov

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2021

I dedicate my thesis to my grandfather who is not with us right now.

Table of Contents

List of Tables

List of Figures

Abstract

With the growing use of the Internet of Things (IoT), the IoT platforms, and the solutions and services related to them, cybersecurity stays of utmost importance. The cyber and interconnected nature of IoT devices makes them vulnerable to various types of cyber-attacks as well as data falsifications. In this thesis, I design and implement a data-driven unsupervised learning approach for anomaly detection for IoT devices such as smart meters, and intelligent electronic devices in power generators. To this end, I employ and evaluate Hierarchical Self-Organizing Maps on real-world smart meter and power system data collected in the USA and Europe. Results show that different types of anomalies could be detected with an F1-score between 0.857 and 0.980 based on the dataset and the type of attack observed.

Acknowledgements

I would like to express my deepest appreciation to my supervisor, Dr. Nur Zincir-Heywood whose expertise, guidance, and continued support made this thesis possible. I am deeply indebted to you for all your help and patience during my studies at Dalhousie University.

I would also like to extend my deepest gratitude to my parents who are the reasons I have come this far.

I am also extremely grateful to Yulduz Khodjaeva who always provided encouragement and unwavering support from my first days in Canada.

Moreover, I wish to thank my friends Akmaljon Jalilov, Akhrorjon Aliev, Behzodbek Malikov, Bekzod Abdullaev, Shakhboz Abdulazizov, Khurshidbek Saidmetov whose help cannot be underestimated.

Chapter 1

Introduction

The Internet of Things (IoT), is a system of interconnected devices with sensors, software, and other technologies that exchange data over communication networks without human-to-human or human-to-computer interaction. The number of IoT devices is increasing so rapidly, it is predicted that the share of Machine-To-Machine (M2M) connections will make up 50 percent of globally connected devices or 14.7 billion in 2023 compared to 33 percent in 2018 (Figure 1.1) [7].

The poor security and interconnected nature of IoT devices make them vulnerable to different types of attacks. Devices without strong secure defense system or software can be used as potential entry points for cyberattacks and data falsification or can cause data exposure [34]. Therefore, as IoT devices become a common part of people's daily lives, cybersecurity stays the biggest concern.

One of the common uses of IoT devices is as smart meters in Advanced Metering Infrastructures (AMIs) [29]. AMI is responsible for collecting consumers' energy consumption data using smart meters, enabling a two-way flow of information between consumers and utility providers. On the one hand, AMI allows commands to be sent to houses for multiple purposes such as real-time pricing, demand-response actions, and disconnecting remotely. On the other hand, smart meter data can be falsified by users/hackers for different reasons from lowering the electricity bills to electricity theft and others [15]. Additionally, organized adversaries can target not only one but multiple smart meters for even bigger data falsification attacks [41, 3]. For example, *additive* data falsification attacks result in high electricity bills and victimize the users. *Deductive*[1] data falsification attacks result in electricity thefts and victimize the electricity provider. Moreover, adversaries can perform both additive and deductive attacks which makes detection challenging for systems that use average aggregates. This is called *camouflage* mode of data falsification [3].

[1]The naming convention is taken from [3]. The authors used it for "subtraction/deducting" attacks.

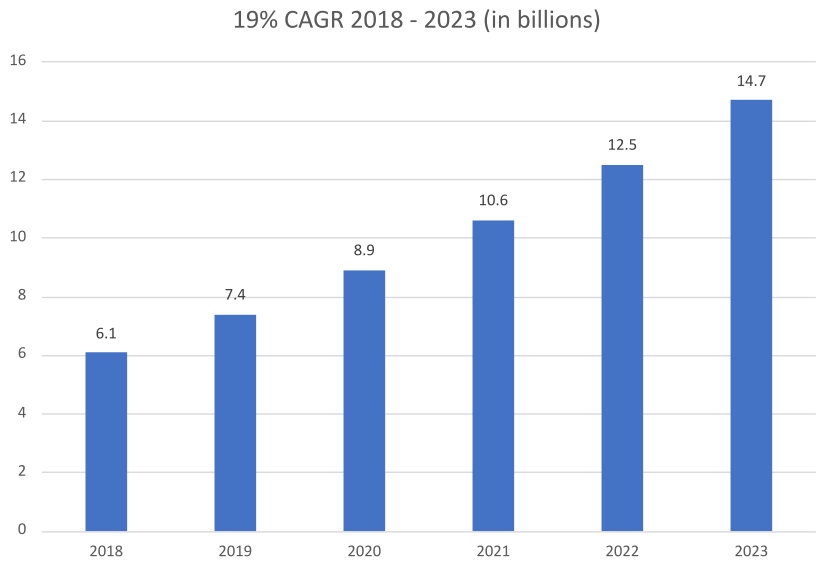


Figure 1.1: The number of Machine-to-Machine connections projected by Cisco over a 5-year period

Another use of IoT devices is power systems. Power systems can contain power generators, breakers, and intelligent electronic devices (IED) that switch breakers on or off. The IEDs relay information back to the supervisory control room. If the framework does not have a well-secured defense system, an attacker can gain access to the network, i.e. the power grid, and perform attacks by sending commands to IED which could cause breakers to open. Besides, the attacker might be able to falsify the energy meter reading or parameters such as current, voltage, sequence, referred to as *Data Injection Attacks* [4]

In this thesis, I explore an unsupervised learning-based approach to detect different types of attacks in the aforementioned IoT systems. To achieve this, I propose a Hierarchical Self Organizing Map (SOM) based approach [16], where I study the effects of temporal information by using explicit time information in the form of timestamps as well as a sliding window with window overlaps. I evaluate the proposed approach with:

1. A small number of features in real-world datasets recorded from houses in New York (New York, US), Austin (Texas, US), Dublin (Ireland) as a part of Pecan Street Project. These datasets consist of real-world electricity consumption data from 2018 and 2019. Since they do not contain any attacks, the attack injection

process is simulated before the implementation of the proposed approach. Injected attack types are divided into three categories for these datasets: Additive, deductive, and camouflage.

2. A large number of features in power system attack datasets from the Mississippi State University in collaboration with Oak Ridge National Laboratory. In this case, the dataset contains data from a power system framework. Attack scenarios are built and simulated with the assumption that an attacker has already gained access to the substation network and can issue commands from substation switches.

Results show that the proposed three-layer Hierarchical SOM model achieves F1-scores between 0.857 and 0.980 based on the type of attacks.

The rest of the thesis is organized as follows: Chapter 2 reviews the literature in this field. Chapter 3 introduces the proposed approach and discusses the datasets, data pre-processing, and the Self-Organizing Map model used as an anomaly detector. Chapter 4 presents conducted evaluations and the results achieved. Finally, conclusions and future work are discussed in Chapter 5.

Chapter 2

Literature Review

The IoT devices have been used in many fields including, but not limited to smart homes, energy, retail, transportation, manufacturing, health, agriculture, and so on [10]. The huge scale of IoT devices requires extensive research to provide reliable security for users. Before coming up with a model for the thesis, it was necessary to analyze existing relevant research in the area. The methodology and results from other works not only give information about existing defense mechanisms but also make it clear why using unsupervised learning techniques, including Self-Organizing Maps can be more useful in terms of IoT security against rapidly evolving data falsification attacks. This chapter summarizes the related IoT cybersecurity literature from the perspective of machine learning approaches used. Thus, supervised learning approaches used for IoT attack detection are summarized in Section 2.1, while unsupervised learning approaches used are discussed in Section 2.2. Lastly, the use of other approaches in the security of IoT is given in Section 2.3.

2.1 Supervised learning approaches

The techniques of supervised learning have been discussed in many studies in order to build enhanced defense models for IoT systems. In [15] Jokar et al. proposed a consumption pattern-based energy theft detector using one-class and multi-class Support Vector Machines (SVM) by addressing the problem of imbalanced data. After dimensionality reduction and normalization, they used the k-means algorithm on a benign dataset to exclude the legitimate abnormal data that might have occurred due to non-malicious actions or events such as seasonality, change of appliances, different usage during weekdays compared to weekends, holidays, etc. Since their dataset does not contain energy theft simulation of the malicious data sample is performed. For training the classifier they used SVM claiming its superior performance in many applications compared to other methods such as neural networks or a likelihood ratio

test. In the first experiment one-class SVM was employed for training using normal samples only, while in the second experiment, multi-class SVM was utilized with benign and malicious data samples. The detection rate reached 76% with a 29% of false positive rate when using one-class SVM. In contrast, multi-class SVM yielded significantly better results: 94% detection and 11% false positive rates.

Another study by Jindal et al. in [14] examined a combination of SVM classifier with the result-oriented Decision Tree (DT) based approach. The model had two steps: During the first step, the parameters (Number of appliances, number of persons, temperature, season, time slot, etc.) of the input is fed into DT and expected output (Expected electricity consumption) is generated. In the second step, previously defined parameters along with actual labels and the output of DT are fed into SVM. As a result, the consumers are classified as normal and malicious. The scheme was able to identify fraudulent consumers with an accuracy rate of 92.5% with a 5.12% false alarm rate.

A classification-based approach for IoT networks using another supervised learning algorithm, the Random Forest, was discussed in [26] by Maniriho et al. For this work, IoTID20 datasets generated in [42] was utilized. The dataset was collected from IoTID20 Testbed that is a smart home system of IoT devices, namely EZVIZ Wi-Fi Security camera and also other smart home devices like Wi-Fi routers, tablets, laptops, smartphones. Wi-Fi Security camera and SKT NGU were victims, while the rest of the devices were attacking devices. It is worth noting that Maniriho et al. used binary classification datasets, subsets of the original IoTID20 primary dataset. Each of the subsets contained Denial-of-Service (DOS), Man-in-the-Middle(MITM), Scan type of attacks, respectively, and normal data. After data pre-processing and feature selection steps, the Random Forest, which is a tree-based ensemble machine learning algorithm, was applied for training. The results showed that the proposed model achieved precision above 99% for all three types of attacks.

2.2 Unsupervised learning approaches

Unsupervised learning-based techniques have also been widely used in the field of IoT security. In [1], Aligholian et al. evaluated the performance of four unsupervised machine learning methods, namely regression, neural networks, clustering, and

projection-based methods, for abnormality detection on real-world smart meter data. One thing to note is that this work identified different categories of features:

(A) *Load Based Features*. These features are collected directly based on the power consumption of the households. Power consumption of the previous 24 hours, or last week, or last month can be examples of this category.
(B) *Contextual Features*. Time of the day, day of the week, weekends, weekdays, holidays, etc. fall into this category.
(C) *Environmental Features*. Temperature, humidity belong to this category.

Then four unsupervised online abnormality detection methods were implemented:

1. *Load Prediction with Regression (LPBSVR)*. This method works by comparing predicted power consumption with actual consumption. The prediction of power consumption is done using Support Vector Regression (SVR), a regression model that is based on outliers, which is suitable for abnormality detection.
2. *Load Prediction with Neural Network (LPBNN)*. LPBNN is similar to LPBSVR except the prediction is done using Neural Network.
3. *Clustered Based Method*. This method divides the dataset into two clusters, namely *abnormal* and *normal* using cluster-based methods, such as Local Outlier Factor or K-Nearest Neighborhood. However, in this work Isolated forest (IF) cluster base method was used with the aim of reducing computational time and enhancing the detection rate.
4. *Projection Based Method*. In this method, the input space is projected to a subspace using dimensionality reduction. For this, Lightweight Online Detector of Anomalies (LODA) was employed due to its computational efficiency.

The dataset used in their research is smart meter data collected from houses in Austin, Texas, US as a part of from Pecan Street Project [32], two of the datasets I used in this thesis. As a performance metric to compare the proposed methods, Matthews Correlation Coefficient (MCC) was used. The results show that IF had the highest performance with MCC equal to 0.81 when utilizing all features. LODA, LPBSVR and LPBNN had MCCs were equal to 0.54, 0.49, 0.47, respectively. However, based on which features were used, the methods had different accuracy rates. For example, in two cases LPBSVR had an MCC of 0.93, which was the highest compared to other

methods: when only historical data was used, and when historical data along with contextual features were used.

Another anomaly detection model using unsupervised one-class learning methods for home IoT devices was proposed by White et al. [45]. Due to the class imbalance in security monitoring as well as class labeling challenges and also to address unknown attacks, they used One-Class Support Vector Machine (OC-SVM). The OC-SVM is a technique developed from SVM and used to find outliers in a dataset by mapping the input space into a high dimensional space using a kernel function [45, 6]. For the experiment, a regular home IoT network consisting of five different IoT devices, namely Amazon Echo, Amazon Fire Tv, Brother Printer, Netgear Arlo Security Camera, and Home Hive Hub was created. As attacks, two types of DoS traffic patterns were used: TCP SYN Flood and UDP Flood. The results showed that the proposed model achieved F-scores ranging between 91.58% and 99.67% based on the device.

Bhattacharjee et al. proposed a semi-supervised consensus-based trust scoring model that could identify compromised smart meters [3]. A notable part of their work was that they proposed the ratio of harmonic to the arithmetic mean showing that those metrics were more stable and more robust for anomaly detection compared to the mean or median. Their work presented an anomaly-based consensus correction technique that detected the presence and type of smart meter data falsification (additive, deductive, camouflage). Moreover, they identified the smart meters that were injecting the false data. The main contribution [3] was that they were able to identify the anomalies using an unsupervised learning approach and then classify the faulty meters based on a semi-supervised based approach.

2.3 Other studies

Fanlin et al. discussed the security problems and corresponding solutions of smart grids in [9]. Specifically, they introduced the security risks of smart meters and online monitoring technologies. Many works related to data falsification in smart grid systems discussed electricity theft from individual customers [28, 27, 48].

On the other hand, in [13], Jiang et al. examined three types of the AMI energy-theft detection schemes: classification, state estimation, and game theory. They

concluded that although each scheme had its own unique features, state-based detection technique could achieve higher detection and lower false positive rates with the aid of specific monitoring devices. Xia et al. proposed a group testing based heuristic inspection algorithm to detect malicious users in a neighborhood area network of a smart grid [47]. They evaluated their algorithm on the data they generated based on a UCI benchmarking dataset and showed that their algorithm had and advantage of conducting fewer inspection steps.

In [27, 48, 2], researchers used average values (mean, median) of electricity measurements to identify data falsification in smart meters. They argued that traditional measures of central tendency might not be robust for legitimate changes in smart meters, increasing false positive rates.

2.4 Summary

To summarize, previous works proposed a number of different supervised, and unsupervised techniques to detect attacks in IoT networks and compromised smart meter data. Many of the models, discussed in this chapter, analyzed network packages and detected attacks only specific to a selected layer. For the works that discussed smart grids along with smart meter datasets, some of the proposed approaches use average aggregates such as median or mean which is sensitive for even abnormal but legit alterations in the environment. Moreover, they do not propose a solution that can be generalized for different types of IoT devices and systems. Thus, in this thesis, my goal is to propose an approach that could be generalized for various types of IoT data, in particular, power systems and devices with different dimensions using Self-Organizing Maps. To the best of my knowledge, no work has investigated the use of Self-Organizing Maps to identify data falsification in power grid-related IoT devices.

Chapter 3

Methodology

This chapter provides information about the data and the architecture of the proposed model. The section 3.1 introduces the datasets employed from the smart meter data to the power system data, while the data pre-processing and attack simulation steps are discussed in sections 3.2 and 3.3, respectively. Then Section 3.4 introduces the unsupervised learning approach, Self-Organizing Maps, whereas the overall architecture for the proposed approach is presented in 3.5.

3.1 Datasets

Obtaining the right and high-quality data for the problem comes with challenges. First of all, companies are responsible not to share users' data because of privacy concerns. Commercial companies can present data by removing personal information or other sensitive information but they are usually costly. For these reasons, in this thesis, two publicly available (Irish and Power System Attack) and two only-academic-use datasets (New York and Austin) are used. Overall information about parameters for all datasets are presented in Table 3.1.

Table 3.1: An overview of the datasets used

Parameters	**New York**	**Austin**	**Irish**	**Power System**
# instances	441599	873286	157992996	78377
# features	79	79	3	128
Date (mm-yy)	05.19-11.19	01.18-12.18	07.09-12.10	2014
Type	Smart meter	Smart meter	Smart meter	Pow. framework
Completeness	100%	99%	99%	N/A
Contains attack	No	No	No	Yes
Real-world	Yes	Yes	Yes	No
Interval length	15-minute	15-minute	30-minute	N/A

3.1.1 New York dataset

The first dataset is New York 15-minute static time-series dataset from Pecan Street Dataport[1] [32]. It is a subset of the primary New York datasets that contain 1-second, 1-minute, 15-minute datasets. The dataset consists of power consumption data as well as household information from 25 houses that participated in Pecan Street Project. Every house has a smart meter installed recording power consumption and also power generation in case there are power generator devices like solar panels. The number of devices in the houses, the number of rooms, the existence of electric vehicles and different kinds of appliances, such as microwave, stove, air conditioner, aquarium, dishwasher, heater and so on make up household information. The dataset was recorded during six months period, from 00:00 May 1, 2019, to 23:45 November 30, 2019, with 100% completeness across all 15-minute intervals. Completeness means how complete a dataset is for every single 15-minute interval during a given period. If some intervals are missing for even one smart meter, the dataset is not 100% complete.

The frequency of power consumption in New York is shown in Figure 3.1. It is evident from the graph that the range of 100-1000 watts consists of the most frequently recorded power consumption.

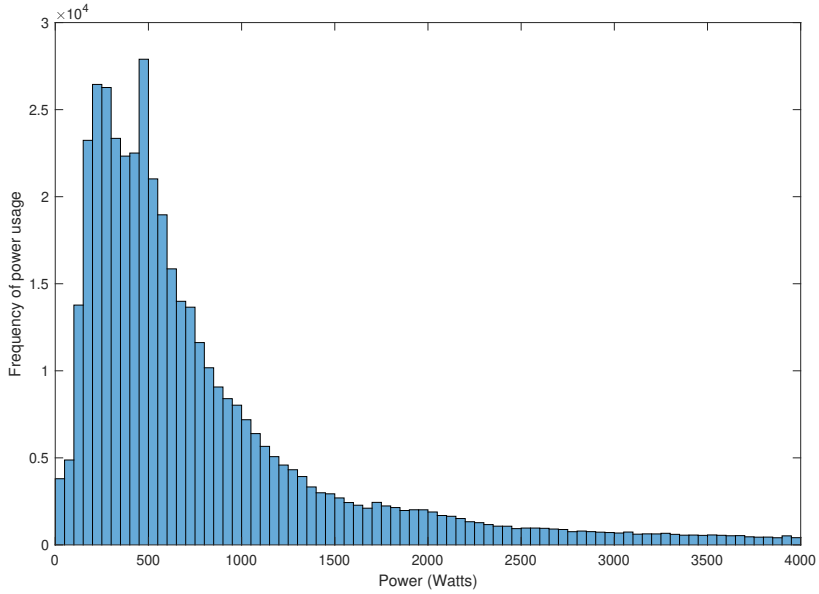


Figure 3.1: New York dataset power consumption frequency

[1]Source: Pecan Street Inc. Dataport

3.1.2 Austin dataset

The second dataset is Austin 15-minute static time-series dataset. Like the previous dataset, it is also collected as a part of the Pecan Street Project [32]. The primary Austin dataset contains 1-second, 1-minute, 15-minute datasets but for this thesis, only the last one (15-minute dataset) is used. Original Austin dataset has information about power consumption, generation, and household information like the number of rooms, the existence of garage, devices (microwave, stove, air conditioner, fridge, dishwasher, heater, etc.) of 25 houses. The dataset was collected during 1 year period, from 00:00 January 1, 2018, to 23:45 December 31, 2018, with 99% completeness. Figure 3.2 shows the frequency of energy consumption recorded in the dataset.

It can be observed that the distribution of power consumption follows an approximate log-normal distribution. The Figure 3.2 can also support this.

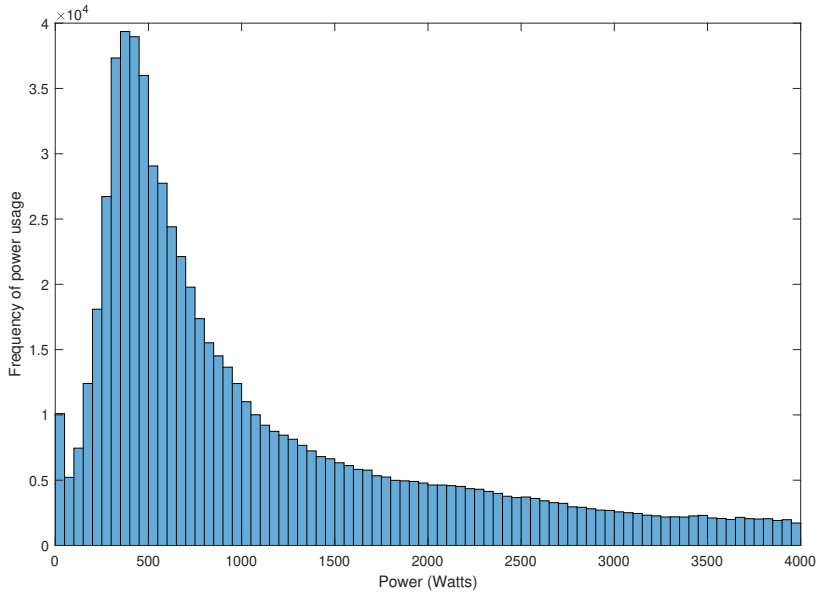


Figure 3.2: Austin dataset power consumption frequency

3.1.3 Irish dataset

The third dataset is Irish 30-minute static time-series dataset[2]. The Commission for Energy Regulation (CER), a regulator for the electricity and natural gas sectors in Ireland, initiated the Smart Metering Project with other organizations in Ireland in

[2]Accessed via the Irish Social Science Data Archive - www.ucd.ie/issda

2007 to assess the performance of Smart Meters [11]. As a part of the project, The Smart Metering Electricity Customer behavior Trials (CBTs) took place during 2009 and 2010 with more than 5000 homes and businesses participating in the project in Ireland. The dataset, recorded from July 14, 2009, to Dec 31, 2010, in anonymized format was made available for further research works. No personal or confidential information is left in the dataset [11]. Like in New York and Austin datasets, the range of 100-1000 watts is the most utilized energy range among the houses being examined in Ireland (see Figure 3.3).

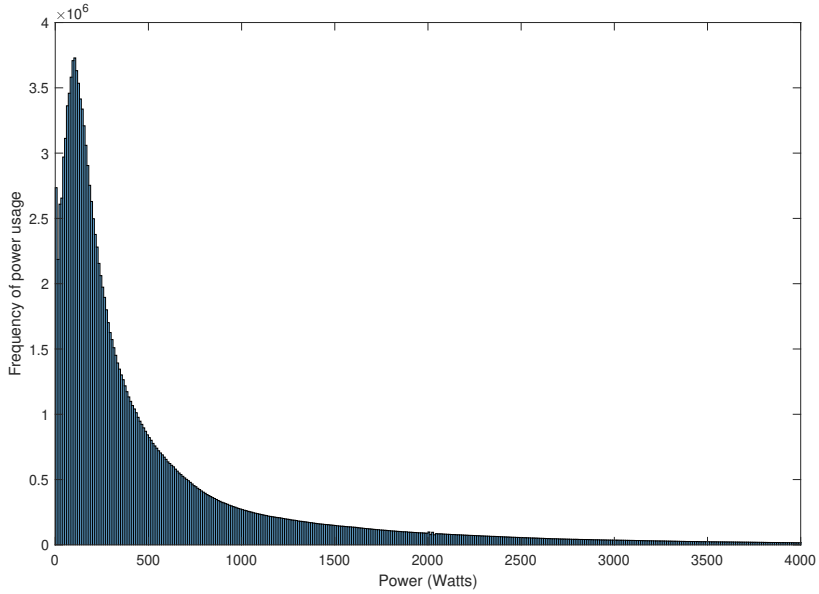


Figure 3.3: Irish dataset power consumption frequency

3.1.4 Power System Attack Dataset

Uttam Ahikari, Shengi Pan, and Tommy Morris from Mississippi State University in collaboration with Raymond Borges and Justin Beaver from Oak Ridge National Laboratories have created 3 datasets using a power system framework [40]. The dataset includes measurements of different electric transmission system behaviors such as normal, disturbance, cyber-attack [43]. Moreover, there are data logs from Snort, a simulated control panel, and relays.

Figure 3.4 shows the power system framework used to create the dataset and simulate the attacks and other abnormal behaviors in the network. The network consists of several components:

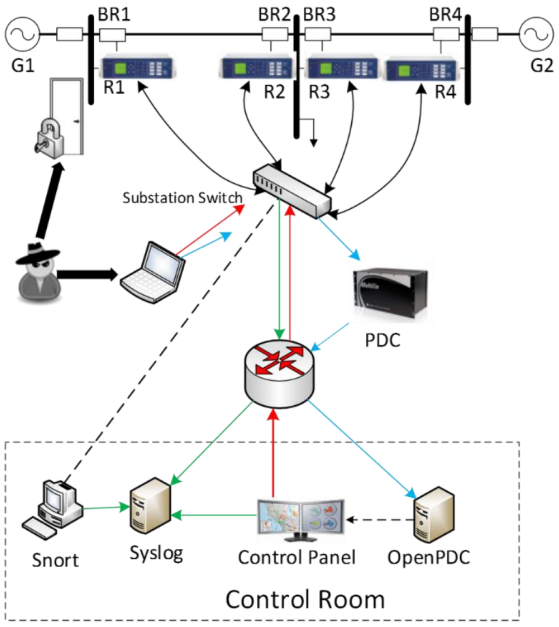


Figure 3.4: Power System Framework

- *Power generators.* G1 and G2 in the diagram are power generators.
- *Breakers.* BR1, BR2, BR3, BR4 are breakers.
- *Intelligent Electronic Devices* (IEDs). The IEDs switch breakers on and off. R1 controls BR1, R2 controls BR2, and so on.
- *Transmission lines.* Line One spans from breaker BR1 to breaker BR2, line Two spans from breaker BR3 to breaker BR4. In other words, line One connects bus B1 to B2, while line Two connects B3 and B4
- *Substation switch and Router.* The IEDs send information through a substation switch through a router to the control room and data acquisition systems.

The IEDs use a distance protection scheme in which turns the breaker on/off detected faults. However, they cannot distinguish if the fault is valid or fake. Operators can also manually send commands to all four IEDs to switch breakers while there is maintenance on the system [4]. Attack and other disturbance scenarios are simulated by manually injecting or issuing commands.

Overall, the dataset contains 128 features. There are 29 types of measurements

from each phasor measurement unit (PMU). A phase measurement unit or synchrophasor is a device that measures the electrical waves on an electrical grid using a common time source for synchronization. For this system, there are four synchrophasors, each giving 29 features. Total 116 features are coming from them. Another 12 features are control panel logs and Snort alerts. The type of all abnormal behavior scenarios, attack events, and the feature list can be found in Appendix A.

3.2 Data pre-processing

Preparing a dataset to employ it with a machine learning model is a crucial part of the process [36, 31, 35]. Even though the datasets used in this thesis have already been prepared for general research purposes, there are still some features/parts that need to be removed or refined to bring the dataset into the ready-to-use format as an input for the proposed model. Since the datasets New York and Austin datasets are much similar and in the same format pre-processing steps for them are almost the same. However, the Irish dataset and Power System Attack dataset need different refinement. The next following three subsections are about data pre-processing.

3.2.1 Pre-processing: New York and Austin datasets

As mentioned in Section 3.1, New York, and Austin datasets had many features including household information about appliances, electric vehicles, rooms, and so on. However, for this thesis only the following features are used:

Smart meter ID. The unique identifier for the smart meter or home is given as an integer value. No conversion is required for this feature.

Timestamp. Time of the data recorded. Given in a string format of *yyyy-mm-dd HH:MM:SS*. This feature is converted into a serial date number that represents the whole and a fractional number of days from a fixed, preset data, in this case, January 0, 0000, in the ISO calendar [39].

Solar1. Power generated by the first solar photo-voltaic system, given in kilowatt (kW). This feature is converted into watts (W).

Solar2. Power generated by the second solar photo-voltaic system, given in kilowatt (kW). This feature is converted into watts (W).

Grid. Electricity used in the given fifteen-minute interval, given in kilowatt. This feature is converted into watts (W).

The rest of the features are removed the dataset. Moreover, since there are three features ($solar1$, $solar2$, $srid$) that make up overall power usage in a given time, it needs to get calculated. For example, $Solar1 = 500W$, $Solar2 = 250W$, $grid = 1000W$, assuming that solar panels have not generated any power, overall power consumption at this time equals to $1000W + 250W + 500W = 1750W$. In general, for each data instance i, we calculate the overall grid meter by $meter(i) = solar1(i) + solar2(i) + grid(i)$. After the grid meter for each data instance is calculated, we no longer use the $solar1$, $solar2$, $grid$ features. As a result, the following three features left: $SmartMeterID$, $timestamp$ and $meter$.

3.2.2 Pre-processing: Irish Dataset

Irish dataset does not contain extra information that needs to be removed, yet given features need processing thoroughly:

Smart meter ID. The unique identifier for the smart meter or home is given as an integer value. No conversion is required for this feature.

Timestamp. Time of the data recorded. Given in a five digit code. Day code is digits 1-3 (day 1 = January 1, 2009), time code digits 4-5 (1-48 for each 30 minutes with 01 = 00:00:00 - 00:29:50, 02 = 00:30:00 - 00:59:59, so on accordingly). This feature is to be converted into serial date number as it is done for *timestamp* feature in New York and Austin datasets.

Grid. Electricity consumed during a 30-minute interval, given in kilowatt. This feature is converted into watts (W).

With these pre-processing steps, all smart grid datasets (New York, Austin, Irish) are to be brought into the same format. It is worth mentioning that the Irish dataset, unlike New York and Austin datasets, contains a much larger number of smart meters.

To be exact, it contains data of 6435 smart meters, while the New York and Austin datasets have 25 smart meters each. Therefore, for the Irish dataset, different number of smart meters will be analyzed: 25, 100, 200.

3.2.3 Pre-processing: Power System Attack dataset

Power System Attack dataset is in a totally different format. Unlike the remaining dataset, this dataset does not contain smart meter data. This dataset has been chosen intentionally to show the robustness and the generalization property of the proposed model. That is, the proposed model can be used in a different types of IoT environments by applying slight changes to datasets.

For this thesis, the 'triple' dataset is used after implementing a few pre-processing phases. This dataset contains all 128 features and labels whose value is one of attack, normal, natural events. Following are the pre-processing steps:

1. The data instances whose label is natural are removed to be able to use in the proposed approach.

2. Initially this dataset is ready to use. However, from this dataset, two more datasets are extracted. The data and features that belong to only one of the IEDs (see Section 3.1.4 for IEDs) are selected from the dataset prepared in step 1 in order to see how it affects the performance of the model.

3.2.4 Sliding window

Lastly, to study the effect of further temporal information, a sliding window approach is employed given that a standard SOM has no capacity to recall the history of patterns directly. To this end, l and k integer numbers are employed such that $0 < k < l$. For $i = 0$, starting from the ith exemplar of a dataset D, we input the data to the SOM using a sliding window approach based on a sliding window of size l, and a window overlap of size k.

This approach is to be used to examine how temporal information affects detection and false alarm rates. To this end, $l = 5$ and $k = 0, 1, 2$ values will be used. These results will be compared to the results where no sliding window approach is used.

3.2.5 Dataset splitting & Feature scaling

Overall, out of four main datasets New York, Austin, Irish, Power System Attack, the following datasets are generated to examine the proposed approach:

- New York additive, New York deductive, New York camouflage.
- Austin additive, Austin deductive, Austin camouflage.
- Additive, Deductive, Camouflage datasets for each of Irish with 25 smart meters, Irish with 100 smart meters, Irish with 200 smart meters (nine Irish datasets, overall).
- Power System Attacks dataset, Power System Attacks dataset with features of one PMU and extra features, Power System Attacks dataset with features of one PMU only.

As the last steps of pre-processing, the following are implemented:

- For a given dataset, 70% of it is used for training, while the remaining 30% is used for testing. The selection is split into training and testing randomly to keep underline data distribution valid for both sets.
- Any data instances that contained undefined numbers are either removed from the dataset or that specific value is assigned to zero.
- Since the SOM learning algorithm is based on Euclidean distances, the range of the variables is very important [37]. If the range of a variable is significantly larger than others, this variable might dominate the map organization [12]. Therefore the values are normalized using Z-score (unit variance) normalization is used [37, 17]. How the algorithm works is shown next.

Z-score normalization

The Z-score normalization method is widely used for normalization in many machine learning algorithms such as Support Vector Machines, artificial neural networks, and logistic regression. It is calculated as:

$$x' = \frac{x - \overline{x}}{\sigma} \tag{3.1}$$

where x is an input vector, $\overline{x} = mean(x)$ is the mean of this feature vector, and σ is its standard deviation.

3.3 Attack Behavior Simulation and Injection

As described in Section 3.1.4, it Power System Attack dataset was generated in a lab environment with different types of attacks. Therefore, for this dataset, or for the one that is extracted from it, no attack simulation is required. They can be used as they are after pre-processing.

On the other hand, as mentioned earlier, the New York, Austin, and Irish datasets are real-world datasets collected as a part of the nationwide projects. These datasets do not contain any data falsification. Therefore, based on [3], we simulate additive, deductive, and camouflage attacks and inject them into the New York dataset. Power consumption data from smart meters can be altered by adversaries or users once they break the defense walls and gain access to the network [28]. Therefore this attack simulation section is entirely about smart meter datasets. Note from Chapter 1, the type of attacks are following:

Additive: Reporting higher than actual power consumption.

Deductive: Reducing actual measured reading of power consumption.

Camouflage: Performing both additive and deductive attacks to evade the detection methods that use mean aggregates [3].

Compromised smart meters

Let N be the number of available smart meters in a dataset, and M be the number of smart meters that adversaries compromise. Then the fraction $S_{mal} = \frac{M}{N}\epsilon(0:1)$ yields the portion of the compromised smart meters. $S_{mal} = 0.30$ means 30% of the smart meters are compromised. In [3], researchers discussed different ranges of $_{mal}$ from 10% to more than 75%. For this thesis, I use the average of this range which is 40%. That is approximately 40% of all smart meters are randomly selected for injecting data falsification attacks.

Data falsification range

Let F_h be an average value for false data that is used to inject attacks (in watts per hour). Some works in the literature used as high as 1200-1500W [21], which could make it easy to detect. In this thesis, $F_h = 800W$ is used to challenge the learning system and study whether the proposed SOM-based approach could work well for low data falsification ranges. Since F_h is 800 watts per hour, the corresponding portion of this amount is falsified based on the dataset. For instance, New York and Austin datasets contain an interval of 15-minute, thus I use an average $F_{avg} = F_h/4$ per data instance when simulating the different types of attacks to the 40% of the smart meters that are randomly selected as described above. For the Irish dataset, on the other hand, around $F_{avg} = F_h/2$ amount is falsified as the dataset consists of a 30-minute dataset so that for all datasets average falsified data to be equal to approximately F_h per hour.

Attack injection

The datasets during different periods of time. Given their recorded date in Table 3.1, we randomly chose a period for each dataset to implement attack simulation. Then only for that period, F_{avg} is falsified for randomly chosen smart meters.

To simulate an *additive* attack, F_{avg} is added. Instead of adding, if F_{avg} is subtracted, then it produces a *deductive* attack. Finally, simulating the camouflage mode of data falsification is more complex compared to additive and deductive attacks. In the camouflage attack mode, for each data exemplar in the dataset that belongs to a compromised smart meter, the following steps are to be taken: Let g_{rand} be a randomly generated real number, $g_{rand} \epsilon [0; 1]$, if $g_{rand} > 0.5$ then F_{avg} is added to this data exemplar, otherwise, F_{avg} is subtracted. When some values are subtracted from actual power consumption to simulate deductive or camouflage attacks, if the new value becomes negative, it has to be changed to zero as the smart meter should not produce negative values.

To sum up, attack simulation has a number of parameters:

- S_{mal} - the ratio of compromised smart meters to all smart meters available.
- F_{avg} - average power consumption value (in watts) to be falsified.

- *Period* - the period in which attack simulation takes place.

3.4 Self-Organizing Maps

A Self-Organizing Map (SOM) or a Self-Organizing Feature Map is an artificial neural network, introduced by the Finnish professor Teuvo Kohonen in the 1980s, that uses unsupervised learning for training to produce a low dimensional representation of the input space [20, 46]. This makes it easier to visualize high-dimensional data. As opposed to error-correction learning such as gradient descent or backpropagation, SOM utilizes competitive learning [18].

An SOM consists of fixed positioned components called neurons or nodes. These components can be arranged as a hexagonal or rectangular grid with two dimensions (Figure 3.5)

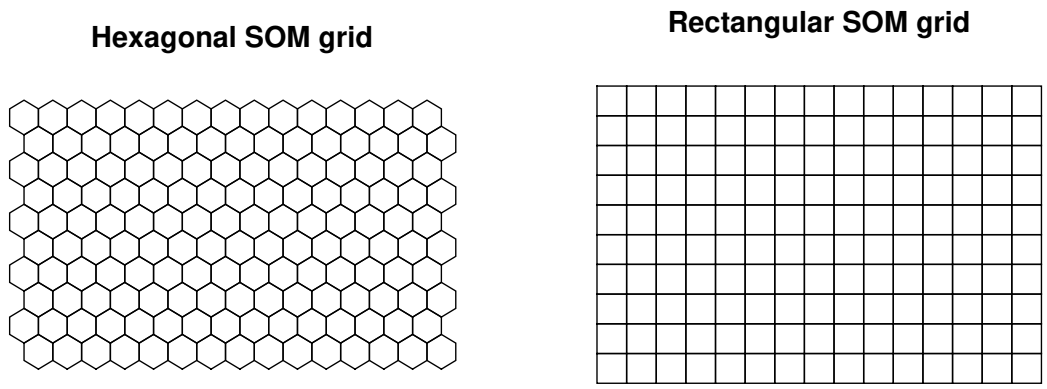


Figure 3.5: Examples of SOM grids with sizes 10×15

Each neuron has a weight vector, that is initialized randomly or linearly randomly. When training data is fed to the SOM network, the Euclidean distance from the input vector to all weight vectors is computed. The neuron that is closest to the input vector is called the Best Matching Unit (BMU)[3]. During training, BMUs and their neighbors are adjusted towards the input vector. With many iterations, the magnitude of the change decreases and the SOM neural network converges to an approximation of the input data distribution. The SOM training is based on two principles:

[3]If a neuron w is a BMU for input vector x, the notion of '*x input vector hits w neuron*' is also used

Competitive learning : the weight vector of a neuron that is most similar to an input vector is modified so that the weight vector becomes more similar to the input vector. This is how the map learns the position of the input data over many iterations.

Cooperative learning : not only the most similar weight vector but its neighbors on the map are also moved towards the input vector. This is how the map self-organizes.

Training of Self-Organizing Maps can be implemented in two ways: sequential and batch training [30]. Both training algorithms are discussed in the next sub-sections.

3.4.1 SOM sequential learning algorithm

Let X be a list of all input vectors $x = [x_1, x_2, ...x_n]^T \in \mathbb{R}^n$ and W be a list of all weight vectors $w_i = [w_{i1}, w_{i1}, ...w_{in}]^T \in \mathbb{R}^n$ of neurons. The SOM learning algorithm for training has several steps as shown the following:

Step (1) Initialize $M_1 \times M_2$ two-dimensional lattice of neurons, and their weight vectors $w_i = [w_{i1}, w_{i1}, ...w_{in}]^T \in \mathbb{R}^n$. Each w_i has a position r_i in the 2D plane. The weight vectors can be initialized randomly or linearly.

Step (2) Randomly pick an input vector x from X.

Step (3) Traverse each neuron of the lattice and calculate the distance between each neuron's weight vector and input vector. For distance measurement, Euclidean distance can be used. That is, $d(x, w_i) = \|x - w_i\|$, where $\|a\| = \sqrt{\sum_{j=1}^{n} {a_j}^2}$. Then the neuron whose weight vector is closest to input vector based on Euclidean distance is identified. This neuron is BMU, w_c, where

$$c = \arg\min_i d(x, w_i) \tag{3.2}$$

Step (4) Weight vectors of the BMU, w_c and its neighbors are adjusted towards input vector:

$$w_i(t+1) = w_i(t) + h_{ci}(t)x - w_i(t) \tag{3.3}$$

where $t \in \mathbb{Z}$ such as $0, 1, 2, ...$ is discrete-time coordinate. Here $h_{ci}(t)$ has an important role: it acts as *neighborhood function*, a smoothing kernel defined

over the lattice points. Neighborhood kernel, used in SOM can be written in terms of the Gaussian function,

$$h_{ci}(t) = \alpha(t) \cdot exp(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}) \tag{3.4}$$

where $\alpha(t)$ is scalar-valued *learning-rate factor* and the parameter $\sigma(t)$ defines the width of the kernel. Both $\alpha(t)$ and $\sigma(t)$ are monotonically decreasing functions of time. r_c and r_i are positions of neurons c and i, respectively [44].

Step(5) Repeat steps 2-4 by a specific number of iterations or until convergence condition is satisfied.

3.4.2 SOM batch learning algorithm

Practically, the SOM batch learning algorithm converges faster and takes less computational cost than sequential learning [22]. For this algorithm, unlike SOM sequential learning algorithm, each input vector in the input dataset is traversed in Step (2). The whole SOM batch learning algorithm is as follows:

Step (1) This step is the same as Step (1) described in SOM sequential learning algorithm

Step (2) For each SOM neuron i, find all input vectors x whose BMU is w_i.

Step (3) The weight vectors of SOM neurons are updated as follows:

$$w_i(t+1) = \frac{\sum_{j=1}^{N} h_{ci}(t) x_j}{\sum_{j=1}^{N} h_{ci}(t)} \tag{3.5}$$

where N is training set size.

Step(4) Repeat steps 2-3 in two phases: (a) *coarse training* phase with large neighborhood radius $\sigma_{coarse}(t)$ and small number of iterations l_{coarse}. (b) *fine training phase* with small and constant neighborhood radius $\sigma_{fine}(t)$. This phase can have many iterations l_{fine} or is continued until convergence.

3.4.3 Visualization using SOM

Being able to see the relationships and the patterns in a dataset can help understand the behavior of the input data. However, we usually visualize things in two or three dimensions. Many existing datasets or the ones to be collected in the future can have high dimensions which make it challenging to analyze the data. Self-Organizing Maps, on the other hand, can be used for exploratory data analysis and visualization of high dimensional datasets. Visualization is one of the most common uses of SOMs [33]. In the next two sections, simple data is used to show the visualization capabilities of SOM.

Two-dimensional data visualization

Let data be two-dimensional with three clusters (Figure 3.6). SOM lattice of size 15×15 is initialized linearly. Figure 3.6a shows the input data and SOM lattice before training. Figure 3.6b displays the same SOM lattice after training. It is clear that lattice now represents input data. The graph also shows that the distances between neighboring units in SOM neurons were almost the same since it was linearly initialized before the training process. However, after the training, distances between neighboring units change. That is, the neurons that are close to clusters are close to each other, while the neurons that do not lay on input data are far from each other. This information can be confirmed in the Figure 3.8

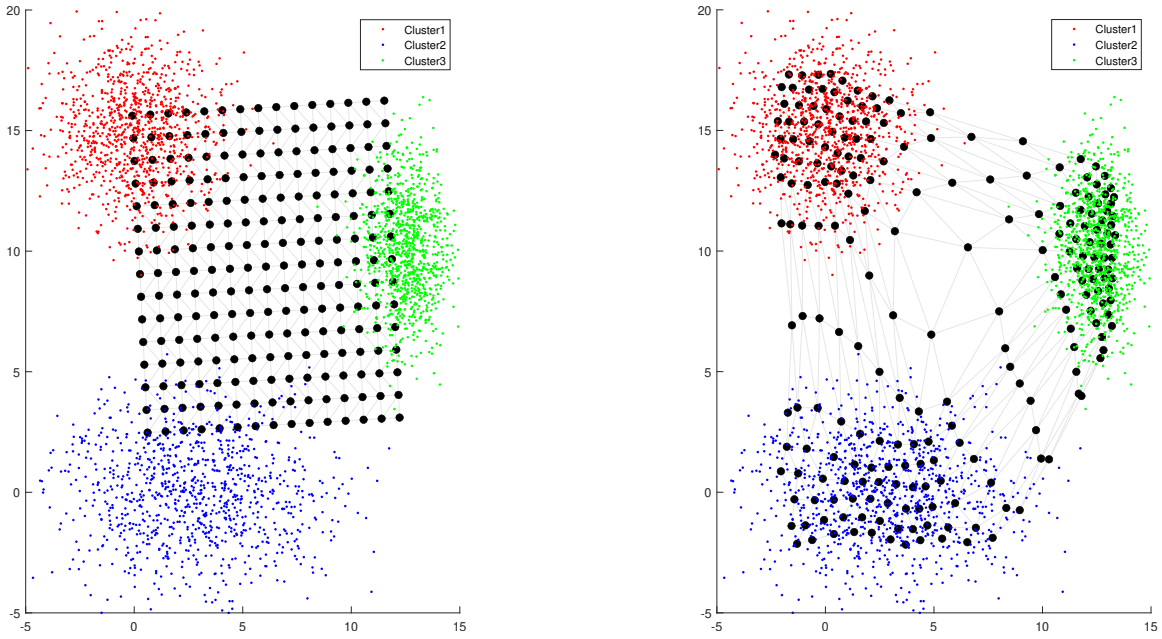
Three-dimensional data visualization

Here, I use three-dimensional data with three clusters (Figure 3.7). Again, the size of the SOM lattice is 15×15. Figure 3.7a demonstrates the SOM lattice with input data before training. After the batch training method, the neurons on the SOM lattice get closer to the input vectors.

Distance matrix

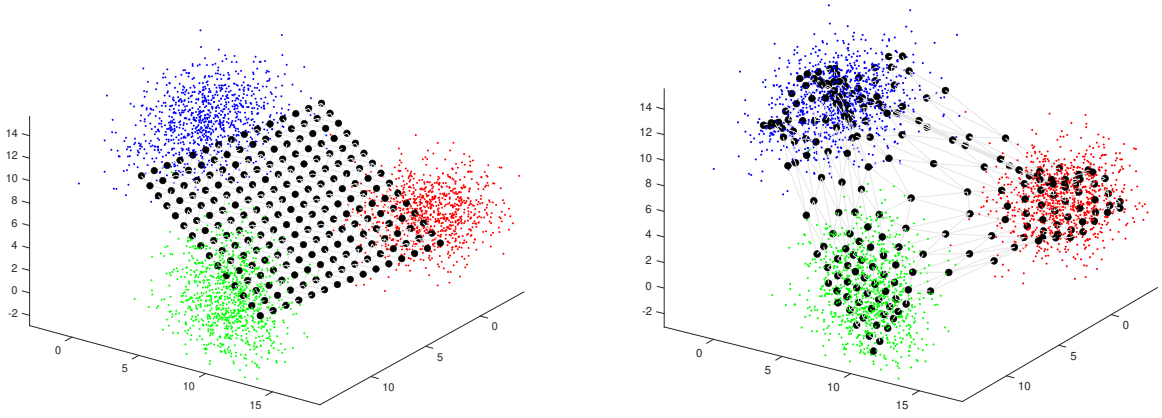
Distances between neighboring units are shown in a distance matrix, or U-matrix[4]. Therefore, the distance matrix visualizes the cluster structure of the map. Figure

[4]U-matrix is a graphical visualization to illustrate the degree of clustering tendency on the SOM via distances between SOM neurons [19, 23].



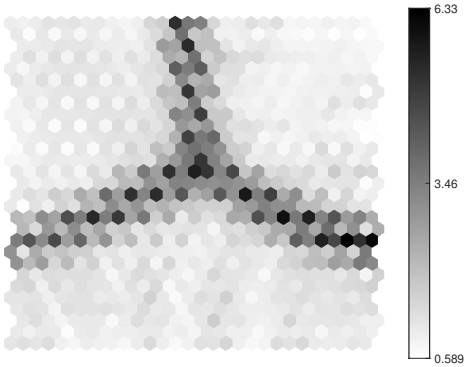
(a) Before training (b) After training

Figure 3.6: SOM 2D map before and after training

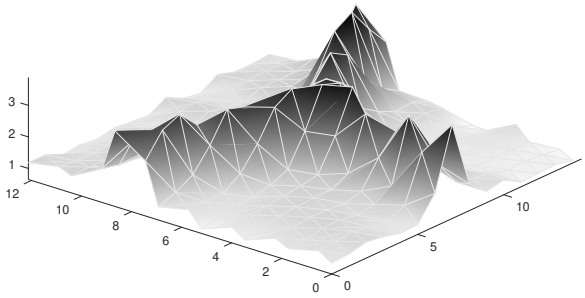


(a) Before training (b) After training

Figure 3.7: SOM 3D map before and after training

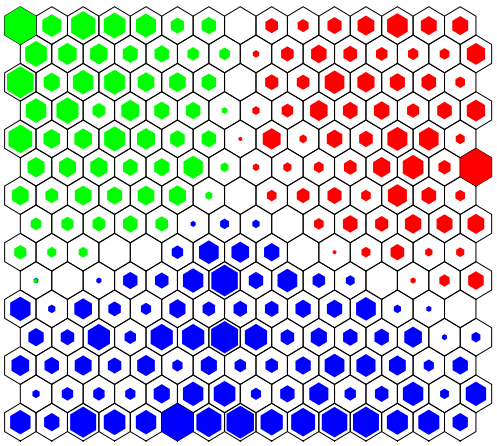


(a) 2D distance matrix

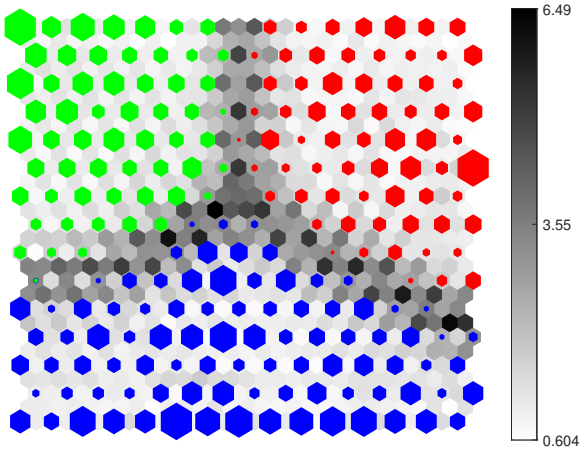


(b) 3D distance matrix

Figure 3.8: SOM - Distance matrix (U-matrix)



(a) Data distribution on trained SOM



(b) U-matrix with data distribution

Figure 3.9: SOM - Data distribution

3.8 shows distance matrix. High values on the distance matrix mean large distances between neighboring units on the map, thus presenting cluster borders. Clusters are usually uniform areas of low values (Figure 3.8b). It is evident from Figure 3.8a that there are three clusters.

Data distribution can also be visualized on SOM nodes, as shown in Figure 3.9a. The bigger the coloring is inside a node, the more training data instances *hit* this node. In other words, the more input data the node is BMU for, the coloring also gets bigger accordingly. Figure 3.9b demonstrates distance matrix with data distribution on it. As is shown, higher values of the distance matrix act as 'walls' between clusters.

3.4.4 Hierarchical Self-Organizing Maps

In earlier sections, detailed information about SOM and the algorithms it uses is given. The input data and examples are shown in the previous section have been created as 'perfectly clustered' for visualizations to be easily understandable. However, when using real-world datasets, in many cases, a neuron might get hit by input data that belong to different clusters. For cases where one neuron is a BMU for input vectors from different clusters, *Hierarchical Self-Organizing Maps* can be utilized. In this section, Hierarchical SOM is explained.

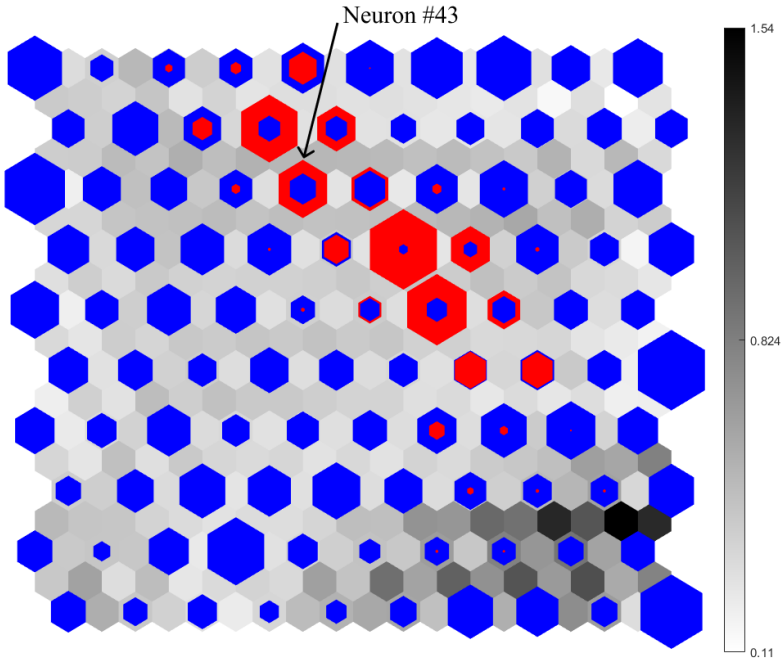
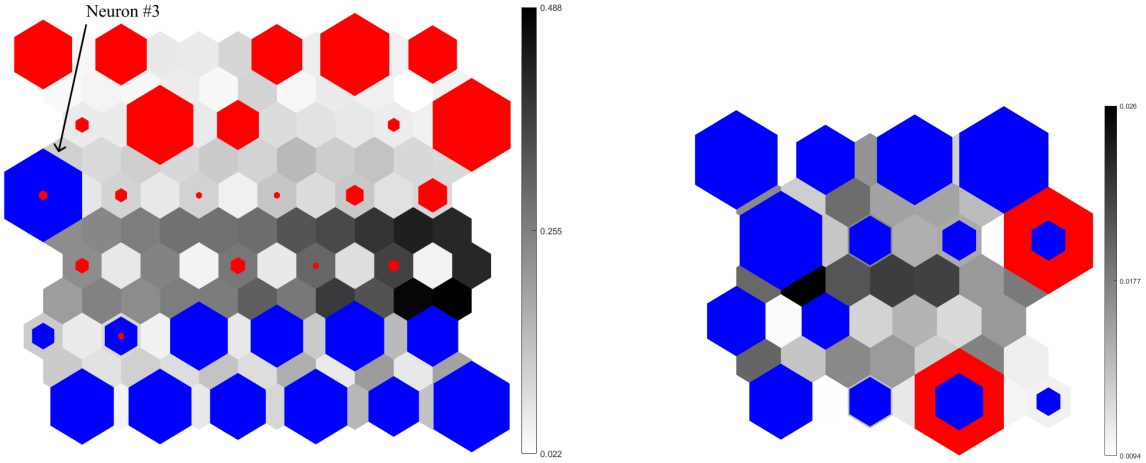


Figure 3.10: SOM layer 1

The basic idea of Hierarchical SOM is to create upper layer maps for some neurons so that a better separation of data is achieved. For training upper layer maps, only the corresponding current neuron's data is used, i.e. a subset of the original dataset. This decreases not only the training time of upper layer SOMs but also enables parallel computing.

Here I use a simple dataset as an example. For a given training dataset, an SOM map of size $M_1 \times M_2$ is created. Let training dataset have 10000 instances, and let $M_1 = 10, M_2 = 10$. In this case, there are 100 neurons (nodes). After training the map, each training instance will have BMU from a list of neurons. Some neurons



(a) SOM layer 2 (b) SOM layer 3

Figure 3.11: SOM - 2nd and 3rd layer maps

will be BMU for input vectors from one cluster class only. However, some neurons will be BMU for differently labeled training instances. For a neuron that is hit by multiple labels, an upper layer SOM with a smaller size is created. To train an upper layer SOM, only the training instances that hit this neuron are used. The purpose of doing so is to keep creating upper layer SOM maps so that differently labeled data are separated from one another.

Figure 3.10 shows that most of the neurons have been hit by clusters shown in blue but neuron-43 has been hit by two different labels, red and blue. Let's assume that training out of the whole training dataset 70 instances (30 red, 40 blue) hit this neuron. These 70 instances are then extracted and used to train upper layer SOM of size 6×6. That is for the second-layer SOM map, the input vector will be 70 input vectors.

Figure 3.11a shows data distribution of 70 input vectors aforementioned on trained second layer SOM. In this case, along with some other neurons, neuron-3 is also hit by both red and blue data. Let's assume 15 input vectors out of 70, have hit neuron-3. Another upper layer SOM of size 4×4 is created for neuron-3 and corresponding input vectors. Then 15 input vectors are extracted from the dataset and used to train third layer SOM.

Creating upper layer SOM maps is continued until a specific condition is met. The conditions can be following:

1. The training instances of different classes being used is totally separated from one another. In this case, the main purpose of creating upper layers is achieved.

2. The maximum number of layers is reached. Creating upper layers can be computationally costly. Based on available resources, a fixed number of layers might be set. For this thesis, I set three for the number of layers. This number is selected empirically.

3.5 Proposed Model Architecture

Datasets to be used in this thesis along with data pre-processing steps, attack simulation, Hierarchical Self-Organizing Maps have been discussed in previous sections. In this section, I will give detailed information about the proposed approach of the thesis. The overall architecture of the proposed approach is presented in Fig. 3.12.

Step (1) Data acquisition.

Step (2) Data pre-processing and feature selection. Since publicly available smart meter datasets do not have attack data in them, the attack simulation process has to be applied to introduce real-world attack scenarios as performed in the literature. Data features are selected as they were provided, no additional data engineering or selection is performed.

Step (3) Learning algorithm training. Hierarchical SOMs are employed and trained as shown in 3.4.4.

Step (4) BMU Labelling post-training. Once the training phase is over, neurons of the SOM are labeled. To this end, after training, each data exemplar from the training set is assigned a post-training BMU based on the closest SOM neuron. Given that the number of neurons in the SOM is less than the training dataset size, each 1st layer training neuron could be a BMU for multiple input vectors. Post-training BMU neurons are labeled based on what class of input vectors hit them:

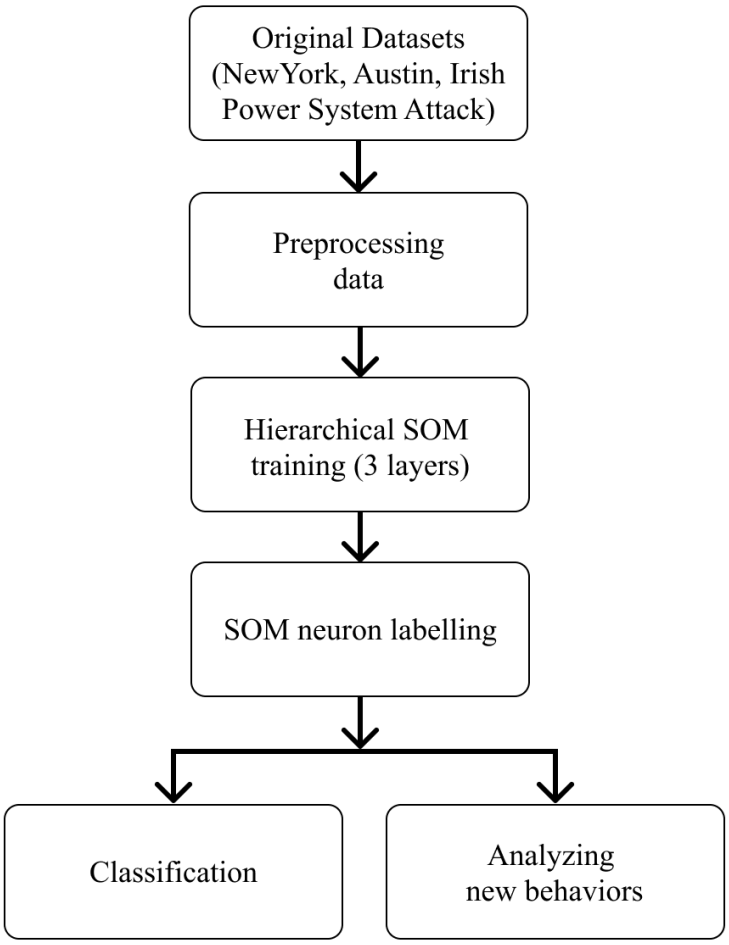


Figure 3.12: The overall architecture of the proposed approach

(a) If a neuron is a BMU for only attack data then that neuron is labeled as an *attack-neuron*.

(b) If a neuron is a BMU for only normal data then that neuron is labeled as a *normal-neuron*.

(c) If a neuron is not a BMU for any input vectors, then it is labeled as a *neutral-neuron*. Such neurons can be used to alert unseen behaviors during the production (test) phase.

(d) Finally, if the neuron is a BMU for both attack and normal data instances, then this neuron is not labeled instead an upper layer SOM is

trained to separate the data points hitting only to this neuron. Therefore an upper layer SOM map will be trained for each such neuron. Step (4) is applied for upper layer SOMs.

For the last upper layer SOM and its neurons, the labeling is slightly different. In this case, for each neuron, if the number of normal data is more than the number of attack data attracted to that BMU, then we label the neuron as a *normal-neuron*, otherwise, it is labeled as an *attack-neuron*. This approach allows setting a threshold in the number of upper layers created for a given neuron. In this thesis, the threshold I used as a proof-of-concept is set to 3.

Step(5) During the testing phase, a test dataset will be input into the trained SOMs. Then, the performance is determined using the labels of the reported BMUs at the end of the test phase. The confusion matrix is calculated based on the correctly detected data instances.

3.6 Summary

To sum up, chapter 3 has provided detailed information about datasets. Then preprocessing steps have been discussed. After that, data falsification, different types of attacks, and attack simulation processes have been explained. After that SOMs and Hierarchical SOMs are introduced. Lastly, the proposed model architecture has been presented. In short, this chapter has been about the methodology steps taken in this thesis. The next chapter is about the evaluation and results of the proposed approach.

Chapter 4

Evaluation and Results

This chapter details implementation parameters, performance metrics, and the results obtained. It should be noted here that to build the Hierarchical SOM architecture, SOM-Toolbox from Aalto University, Finland is employed [12, 19].

4.1 Parameters

There are a number of parameters that directly and indirectly affect the performance of the model. Some of the parameters are chosen based on known practices, while the others are chosen empirically based on experiments performed as part of the research.

SOM parameters are chosen as follows:

Map lattice. For map lattice, the hexagonal lattice is chosen over rectangular lattice because it does not favor horizontal and vertical directions as much as a rectangular grid [18].

Map lattice initialization. Linear initialization is chosen over random initialization. If random initialization is used, it would take a few hundred initial steps even before the unordered vectors get ordered. On the other hand, when linear initialization is used, the first two eigenvectors of the autocorrelation matrix of input x with the largest eigenvalues are determined, then these eigenvectors span a two-dimensional linear subspace. As a result, the array of the lattice is defined along this subspace and the main dimensions of the lattice are the same as the top two largest eigenvalues. Therefore, it can be seen in Figure 3.7a that initialized lattice has a centroid coinciding with that of the mean of x.

Training algorithm. For training, the batch algorithm is chosen. As explained in Section 3.4 SOM batch algorithm generally has a faster convergence rate and less computational cost.

Neighborhood function. For the neighborhood kernel, the Gaussian function 3.4 is chosen because of its wide use and smoothness.

Layer 1 map size. Choosing the right map size is important as it has a direct impact on computational cost and performance trade-off. If the map size is too small, then the weight vectors of the neurons might not be able to approximate the input vectors. However, if it is too big, then it will increase the training time. For these reasons, choosing a map size that is proportional to the input data is preferred. So, if the size of layer 1 map is $M_1 \times M_2$, then $M_1 = M_2 = 2\sqrt[6]{N}$. These SOM sizes are identified empirically by using different map sizes to select the best-performing ones.

Layer 2 map size. Layer 2 map should be smaller than layer 1 as the number of input data instances for layer 2 is much smaller than the size of the original input data. So, it's $M_1^2 \times M_2^2$ where $M_1^2 = M_2^2 = M_1/2$

Layer 3 map size. Layer 3 map size is $M_1^3 \times M_2^3$ where $M_1^3 = M_2^3 = M_1/4$

All parameters of SOM used in this thesis can be found in Table 4.1.

Table 4.1: Parameters used for training the SOM

Parameter	**Value**
Map lattice	Hexagonal
Map lattice initialization	Linear
Training algorithm	Batch
Neighborhood function	Gaussian
Layer 1 map size $M_1 \times M_2$	$M_1 = M_2 = 2\sqrt[6]{N}$
Layer 2 map size $M_1^2 \times M_2^2$	$M_1^2 = M_2^2 = M_1/2$
Layer 3 map size $M_1^3 \times M_2^3$	$M_1^3 = M_2^3 = M_1/4$

As discussed earlier, attack injection is simulated for New York, Austin and Irish datasets. The parameters used for attack simulation is given in Table 4.2, $S_{mal} = 40\%$, that is, 40% of all smart meters are compromised. The average data falsification used is 800 watts per hour.

Table 4.2: Parameters used for smart meter datasets

Parameters	**New York**	**Austin**	**Irish**
S_{mal}	0.4	0.4	0.4
F_h	800W	800W	800W
F_{avg}	200W	200W	400W
Recorded date	05.2019-11.2019	01.2018-12.2018	07.2009-12.2010
Attack period	30 days	30 days	30 days
Number of smart meters	25	25	25, 100, 200

4.2 Performance Metrics

Since the datasets are unbalanced, the performance is measured using Precision (P), Recall (R), and F1-score. Let *TP*, *TN*, *FP*, and *FN* be True Positive, True Negative, False Positive, and False Negative, respectively. Then Precision, Recall, and F1-scores are measured as follows:

$$P = \frac{TP}{TP + FP} \tag{4.1}$$

$$R = \frac{TP}{TP + FN} \tag{4.2}$$

$$F1 = \frac{2PR}{P + R} \tag{4.3}$$

These metrics give a better evaluation of the performance of the proposed system for imbalanced data over accuracy rate which is calculated as shown in Equation 4.4.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

4.3 Training phase

Based on the parameters selected in Section 4.1, the training takes place using SOM on the data. In this section, I will show how data distribution looks on the first layer SOM with the examples of *the New York additive attack dataset* and *Power System Attacks dataset*. New York additive attack dataset contains normal data and attack data that have been injected as additive attacks during the pre-processing step.

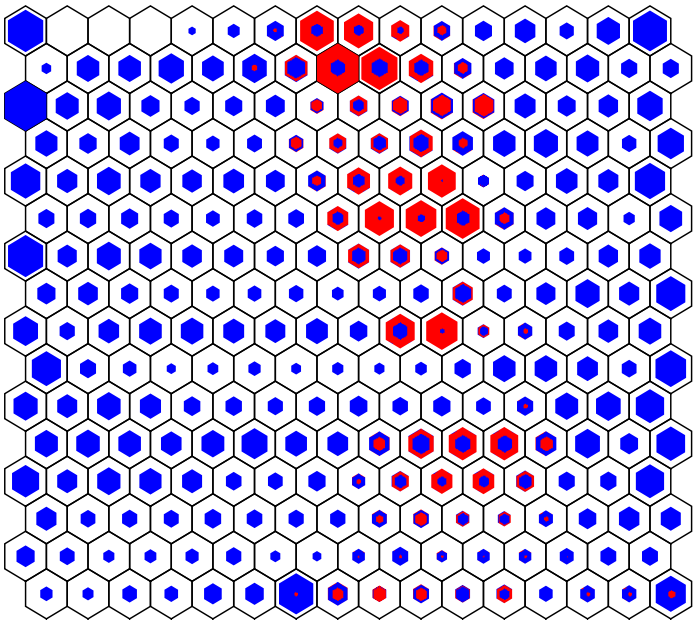


Figure 4.1: Data distribution on the trained SOM – New York additive attack dataset

Figure 4.1 shows New York data distribution on trained SOM. One can see that majority of the neurons are hit by only normal data while others are hit by both attack and normal data instances. For this particular example, the advantage of the Hierarchical SOM becomes clear. If there were no upper layer maps, false positive and false-negative rates would have increased for these *'red-blue'* neurons. However, in the Hierarchical SOM model proposed in the thesis, for each of these neurons, upper layer maps will be built and normal and attack inputs can potentially be separated from each other.

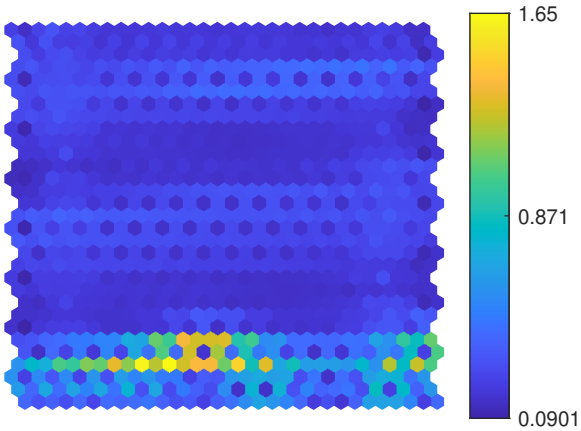


Figure 4.2: 2D distance matrix – New York dataset with additive attack

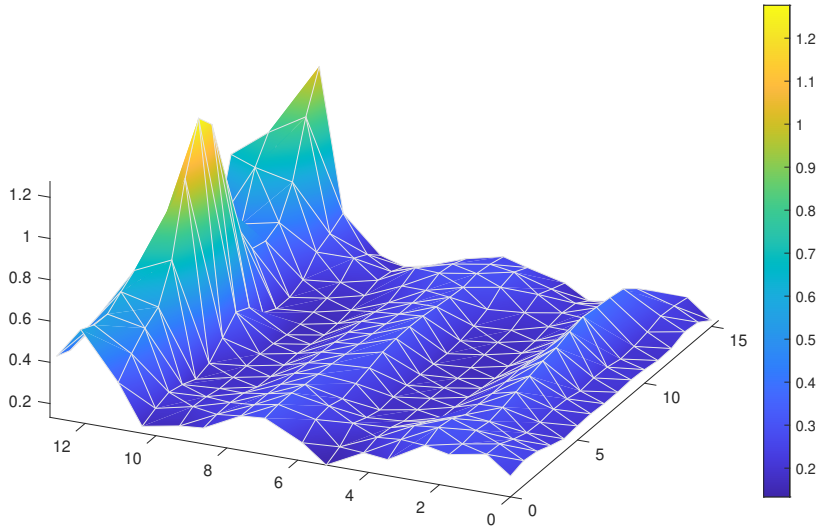


Figure 4.3: 3D distance matrix – New York dataset with additive attack

Distance matrix (average distances between neighboring vectors of the map) is shown in Figure 4.2 and 4.3. Large distances between neighboring vectors represent cluster borders and thus "cluster landscape" is formed over the SOM. On a 2D graph, the coloring code can help to see cluster borders. The darker blue the color is the shorter the distances. Vice versa, as the color changes toward yellow, the distances get larger. Based on the visualization in Figure 4.2, there seem to be four clusters in the data. When a 3D model of the distance matrix is used, it is easier to see the borders.

In order to analyze the SOM for high dimensional data, the Power System Attacks dataset is studied. Figure 4.4 shows the input data distribution on SOM nodes after training the first layer map. It is evident from the graph that attacks have not been split from normal data which creates the need for the second layer SOMs, Figure 4.5. It can also be confirmed in Figure 4.6 where a post-training 3D distance matrix is provided.

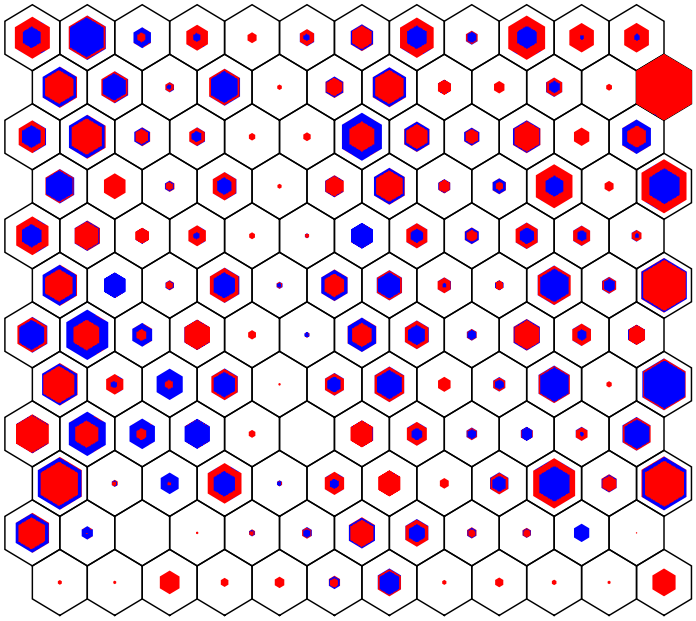


Figure 4.4: Data distribution on the trained SOM – Power System Attacks

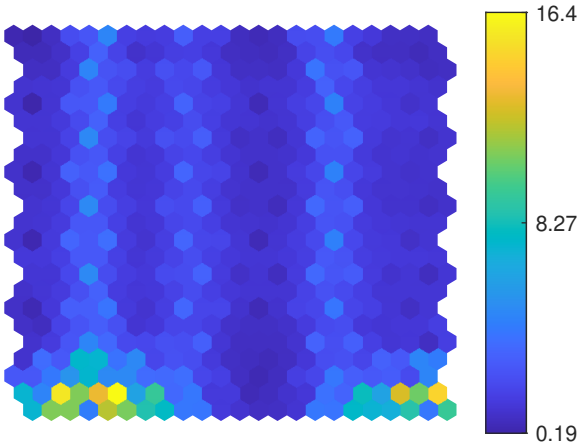


Figure 4.5: 2D distance matrix – Power System Attacks

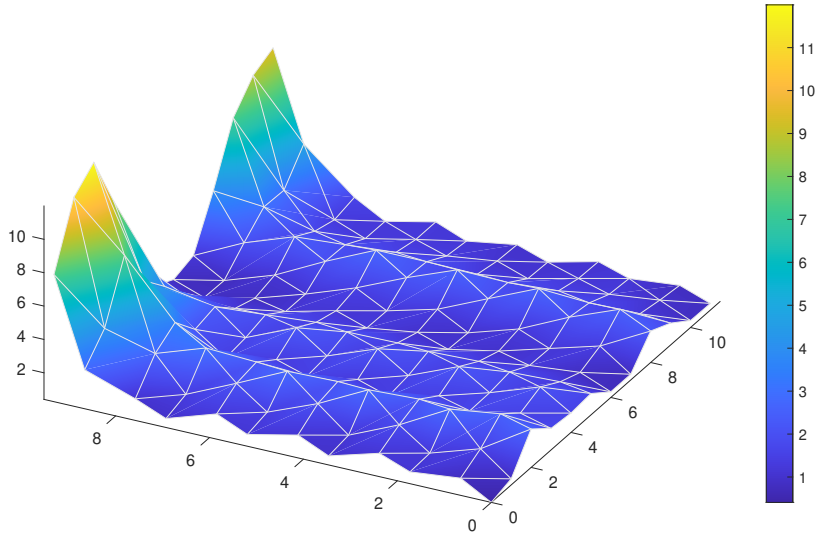


Figure 4.6: 3D distance matrix – Power System Attacks dataset

4.4 Results

In this section, evaluations and results are discussed. Recall from Section 3.2, for each of New York, Austin and Irish datasets three types of attacks, namely additive, deductive, camouflage are injected into the data. Table 4.3 shows the performance (Precision, Recall, F1-score) of the proposed Hierarchical SOM model on the New York dataset. For the additive attack dataset, the precision, recall, and F1-scores are around 0.965. Performance is even better for deductive attacks with 0.980. F1-score is 0.946 for camouflage data falsification. Even though camouflage attacks are challenging to detect, the proposed model achieves over 0.945.

Table 4.3: The results of the proposed model on the New York dataset

Attack type	**Precision**	**Recall**	**F1-score**
Additive	0.965	0.964	0.965
Deductive	0.984	0.976	0.980
Camouflage	0.955	0.936	0.946

Table 4.4 indicates the detection performance on the Austin dataset. Like the New York dataset, the best detection rate is achieved when the deductive attack dataset with 0.946 For additive and camouflage attack datasets the F1-scores were almost equal (0.932). Another common behavior of the proposed model on both New York and Austin datasets is that the precision was the lowest for camouflage attacks.

Table 4.4: The results of the proposed model on the Austin dataset

Attack type	**Precision**	**Recall**	**F1-score**
Additive	0.940	0.925	0.932
Deductive	0.956	0.935	0.946
Camouflage	0.932	0.932	0.932

Since there are three subsets of the original Irish dataset, where 25, 100, and 200 smart meters' data are randomly selected, the results are put into one table for better analysis. In the first case, the Irish dataset with 25 smart meters is used. F1-score for additive attack detection is 0.962. For the deductive attack dataset, one can see that recall was slightly less than 0.9 which means for this dataset, the false-negative rate was higher than 0.1. For the camouflage dataset, the performance is better (0.929) than the one for the deductive dataset. In the second case, the Irish dataset with 100

smart meters is examined. F1-scores for additive, deductive and camouflage datasets were 0.933, 0.907, 0.857, respectively. Lastly, a bigger dataset, the Irish dataset with 200 smart meters is utilized. For all datasets, different attacks have been injected as done previously. The performances are almost the same: around 0.870. In a bigger picture, it can be seen that the performance of the proposed model works well in different scales, as well.

Table 4.5: The results of the proposed model on the Irish dataset

Dataset	**Attack type**	**Precision**	**Recall**	**F1-score**
Irish (25 smart meters)	Additive	0.967	0.957	0.962
	Deductive	0.940	0.885	0.912
	Camouflage	0.946	0.912	0.929
Irish (100 smart meters	Additive	0.938	0.927	0.933
	Deductive	0.930	0.885	0.907
	Camouflage	0.887	0.829	0.857
Irish (200 smart meters)	Additive	0.892	0.863	0.877
	Deductive	0.909	0.848	0.877
	Camouflage	0.891	0.850	0.870

Power System Attacks (PSA) dataset is a different IoT system related to a power grid. As shown in Section 3.2 the dataset has 128 features. To test the performance of the proposed model three forms of this dataset have been used (Table 4.6):

PSA Full dataset. The dataset has all 128 features and all given instances. Precision, Recall and F1-score are 0.974, 0.981, 0.977, respectively.

PSA One PMU with additional features dataset. As explained in Section 3.1.4, in this dataset the power system framework has four intelligent electronic devices that control breakers and relay information back to the control room. Four phase measurement units (PMUs) measures 116 measurements or features in the dataset. Apart from the feature from phase measurement units, there are 12 extra features as well. PSA One PMU with additional features dataset contains 29 features from one PMU including the extra features. For this dataset, the performance has reduced by only 0.001 compared to the previous evaluations. However, the precision stays the same.

PSA One PMU dataset. In this dataset, only one PMU is considered and therefore

extra features have been removed. The proposed model still has been able to achieve high performance with F1-score of 0.977.

Table 4.6: The results of the proposed model on the Power System dataset

Dataset	**Precision**	**Recall**	**F1-score**
PSA Full	0.974	0.981	0.977
PSA One PMU with additional features	0.974	0.979	0.976
PSA One PMU	0.972	0.982	0.977

Even though the IoT system used to collect the Power System Attacks dataset is totally different from the smart grid system of the New York, Austin, and Irish datasets, the proposed model has had high detection rates of attacks. This confirms that the model can be generalized and expanded to a different IoT environment.

4.5 Analyzing sliding windows with window overlaps

As a part of the experiments conducted for this thesis, a sliding window approach is utilized using window overlaps for exploring the effects of implicit temporal information in data representation. The aim is to explore how this approach can affect performance compared to explicit timestamps to represent temporal information. Table 4.7 shows the performance on all smart meters dataset with a sliding window, where l equals 5, and k equals 0. For any dataset, the performance has decreased. For New York additive, deductive, and camouflage attack datasets, the performance is better than others with 0.801, 0.832, 0.808 F1-scores. All Austin datasets have F1-scores around 0.71-0.73. The sliding window approach affected the Irish dataset with 25 smart meters. For the Irish (25) deductive attack dataset, overall performance is 0.500. However, for the other two Irish datasets where 100 and 200 smart meters are used, the model performed at least as good as approximately 0.670.

The results on the sliding window with $l = 5, k = 1$ are shown in Table 4.8. In general, the overall performance is slightly improved in this case with only three exceptions (New York additive, New York deductive, and Irish (100) camouflage). The sliding window with $l = 5, k = 2$ (Table 4.9) has not shown a significant improvement over the previous one.

Table 4.7: Evaluations of the SOM using a Sliding window: l = 5, k = 0

	Additive			**Deductive**			**Camouflage**		
Dataset	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
New York	0.829	0.776	0.801	0.853	0.812	0.832	0.825	0.792	0.808
Austin	0.747	0.725	0.736	0.759	0.712	0.735	0.744	0.695	0.719
Irish (25)	0.592	0.480	0.530	0.583	0.439	0.500	0.627	0.521	0.569
Irish (100)	0.804	0.753	0.778	0.776	0.683	0.725	0.730	0.670	0.698
Irish (200)	0.739	0.685	0.711	0.778	0.730	0.753	0.746	0.670	0.706

Table 4.8: Evaluations of the SOM using a Sliding window: l = 5, k = 1

	Additive			**Deductive**			**Camouflage**		
Dataset	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
New York	0.815	0.776	0.795	0.834	0.805	0.819	0.825	0.821	0.823
Austin	0.751	0.723	0.736	0.765	0.731	0.747	0.759	0.711	0.734
Irish (25)	0.582	0.507	0.542	0.611	0.506	0.554	0.591	0.486	0.534
Irish (100)	0.802	0.766	0.783	0.765	0.670	0.731	0.720	0.669	0.694
Irish (200)	0.751	0.695	0.722	0.786	0.740	0.762	0.751	0.688	0.718

Table 4.9: Evaluations of the SOM using a Sliding window: l = 5, k = 2

	Additive			**Deductive**			**Camouflage**		
Dataset	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
New York	0.853	0.805	0.828	0.846	0.816	0.831	0.831	0.823	0.827
Austin	0.757	0.733	0.745	0.773	0.727	0.749	0.755	0.701	0.727
Irish (25)	0.606	0.541	0.572	0.612	0.500	0.551	0.609	0.553	0.580
Irish (100)	0.756	0.715	0.735	0.750	0.682	0.715	0.732	0.663	0.696
Irish (200)	0.755	0.704	0.729	0.786	0.749	0.767	0.748	0.683	0.714

The impact of the sliding window approach on the Power System Attacks dataset can be seen in Figure 4.10, where $l = 5, k = 0, 1, 2$. It is clear that there has been an improvement in two datasets (PSA Full dataset and PSA with one PMU only dataset) when window overlap size equals 2 compared to other window overlap sizes. Power System Attacks dataset of type One PMU with extra features had the best performance when $l = 5, k = 1$.

Overall, based on the results shown in the aforementioned tables, and the results given in Section 4.4, the proposed Hierarchical SOM model performs better when no sliding window is used in the presence of explicit time information. However, when no explicit timestamps are available in the data then sliding window approach seems

to work well to represent temporal information. However, further study on this is necessary, since the Power System Attacks dataset was the only which had these characteristics in this thesis.

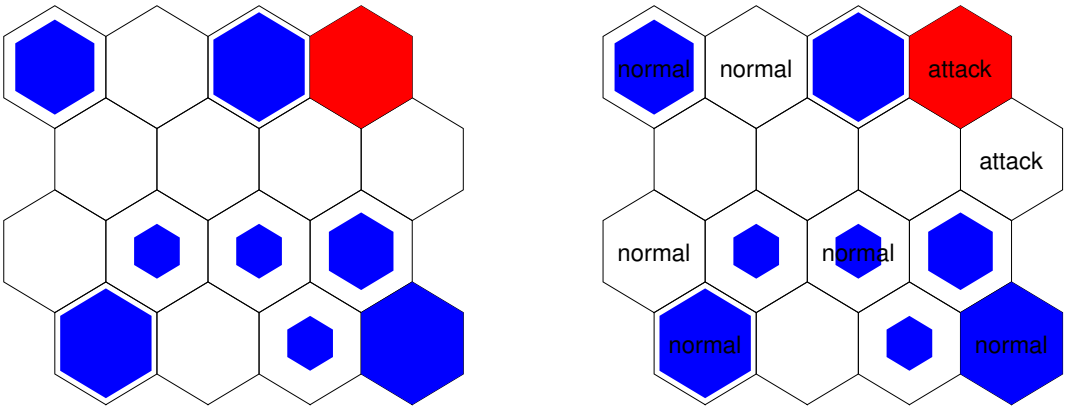
Table 4.10: Evaluations of the SOM using a Sliding window on Power System Attacks datasets

	Dataset	**Precision**	**Recall**	**F1**
Sli. win. l = 5, k = 0	Full	0.924	0.930	0.927
	One PMU extra features	0.921	0.918	0.920
	One PMU only	0.923	0.930	0.927
Sli. win. l = 5, k = 1	Full	0.924	0.925	0.925
	One PMU extra features	0.928	0.940	0.934
	One PMU only	0.925	0.925	0.925
Sli. win. l = 5, k = 2	Full	0.931	0.944	0.937
	One PMU extra features	0.927	0.931	0.928
	One PMU only	0.929	0.934	0.931

4.6 Analyzing new behaviors

As the system we use evolves, the data they produce also evolve. As a result new behaviors both on the normal usage side as well as on the attack side continuously grows. To be able to recognize new behaviors, I leverage the "neutral-neurons" of the proposed Hierarchical SOM approach. Neutral-neurons are the neurons that were not hit by any data during the training phase. In other words, they do not become BMUs for any of the known behaviors (training data). However, it is possible that as the behaviors evolve in the real world, these neurons could be the BMUs for those new (unknown) behaviors. I explored this phenomenon during testing (unknown/new test data). Figure 4.7 shows an example of such a case. This example is taken from a real New York attack dataset. Training data distribution on trained SOM in the third layer is shown in Figure 4.7a. Note that, the attack data hits are shown in red color, normal data hits are shown in blue color, and no data hits are shown with no color. It can be seen that out of 16 neurons, only 1 of them is hit by attack data, while 8 other neurons are hit by normal ones, and the remaining 7 are not hit by any data instances (neutral-neurons).

Figure 4.7b shows how the trained neurons approximate corresponding test data



(a) Training data distribution on SOM neurons on 3rd layer

(b) Actual labels of test dataset on 3rd layer

The n

Figure 4.7: New behavior of input data on trained SOM

on the third layer. It can be seen that some test data hits the neurons that are labeled as an attack or normal after training. The attack-neurons (red-colored) are BMU for some of the attack test data, while four of the normal-neurons (blue-colored) are BMUs for normal test data. However, three neutral-neurons also are BMUs for some data instances. It means that the test input data have new behaviors which are not seen during the training phase.

This information can also be illustrated as in Table 4.11. Neurons are numbered sequentially column-wise starting from the top left. The table gives a more detailed comparison of neuron hits since it provides exactly how much input data hit each neuron. Neuron #1 is hit by 3 training data instances of normal class. So this neuron is to be labeled as 'normal-neuron'. Likewise, neuron #13 is hit by 5 attack training data so this neuron is labeled as 'attack-neuron' and neuron #3 is labeled as 'neutral-neuron'. How these neurons are hit during testing is shown in the last two columns. Neuron #13, an attack-neuron, is hit by one attack instance of test input, while another attack data hits neuron #14, a neutral-neuron.

This analysis shows how SOM maps and their neurons are behaving in each layer and how they can be leveraged to recognize new behaviors. This feature of SOM can further be improved by labeling neutral-neurons based on their closest labeled neuron. This is an example of just one map in the third layer. However, a similar approach could be automated for the analysis of all neutral-neurons (based on training) during testing.

Table 4.11: 3rd layer neuron hits for new behavior analysis

	Training		**Testing**	
Neuron #	**Attack**	**Normal**	**Attack**	**Normal**
1	0	3	0	1
2	0	0	0	0
3	0	0	0	1
4	0	4	0	1
5	0	0	0	1
6	0	0	0	0
7	0	1	0	0
8	0	0	0	0
9	0	4	0	0
10	0	0	0	0
11	0	1	0	1
12	0	1	0	0
13	5	0	1	0
14	0	0	1	0
15	0	2	0	0
16	0	5	0	4

4.7 Data clustering using SOM

In Section 3.4, it was discussed that SOM maps can be used to show the clusters in the data. Visual inspection is helpful when there is a clear separation between clusters as in Figure 3.8a. However, when data is in a higher dimension, the separation might not be clear in the visualizations (Figure 4.8). For such cases, the SOM map needs to be partitioned.

In order to achieve partitioning, the K-means clustering method [24, 38, 25] is used with k = 20. To this end, Davies-Boulding clustering index (DBI) [8] is employed. DBI index becomes minimum with the best clustering. To illustrate this, the first layer map of Power System Attack full dataset given in Figure 4.9. As shown in Figure 4.9a, DBI index becomes minimum, when the number of clusters is 12 (DBI = 0.671 for this case). Figure 4.9b shows these 12 clusters on first layer SOM neurons.

Likewise, the number of clusters on the first layer map can be found for each dataset. The table of datasets is summarized in Table 4.12. A full list of visualizations are given in Appendix

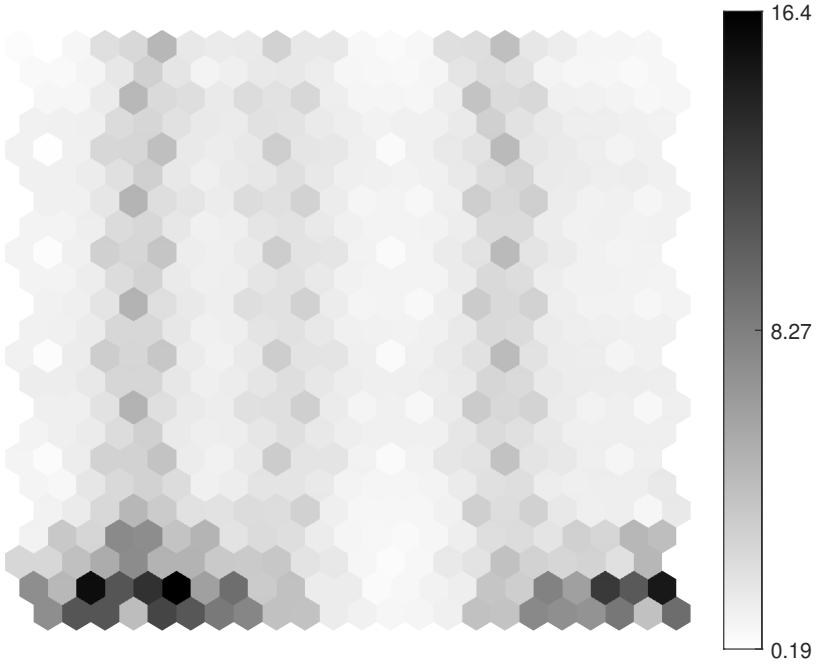
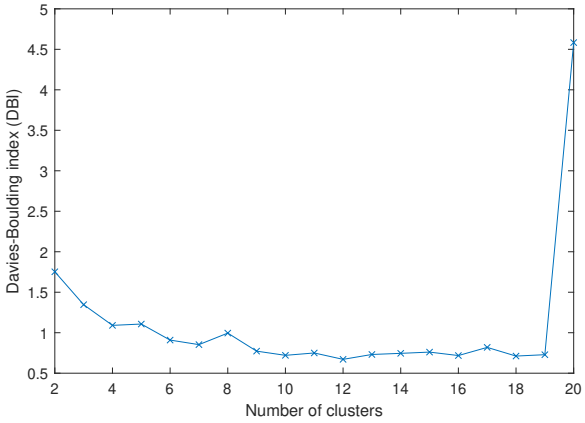
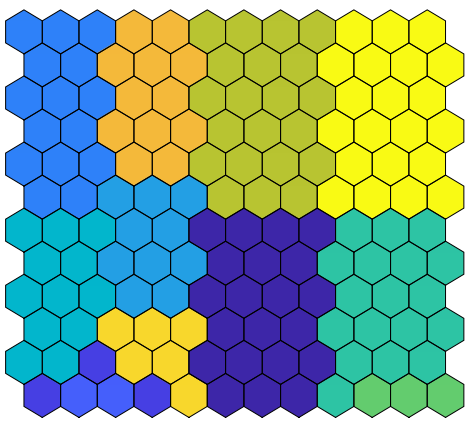


Figure 4.8: Distance matrix - Power System Attack



(a) Davies-Boulding clustering index



(b) Map partitioning

Figure 4.9: Clusters formed on the trained SOM

Table 4.12: The number of clusters found on the 1st layer SOM map

Dataset	**Dataset type**	**Clusters found - 1st layer**
New York	Additive	16
	Deductive	16
	Camouflage	17
Austin	Additive	16
	Deductive	6
	Camouflage	6
Irish (25)	Additive	14
	Deductive	19
	Camouflage	14
Irish (100)	Additive	18
	Deductive	17
	Camouflage	16
Irish (200)	Additive	18
	Deductive	17
	Camouflage	16
PSA	Full	10
	PMU extra feature	11
	PMU only	11

4.8 Discussions

In order to be able to compare the results of this thesis with the results of existing researches, I replicated the works of [3] since they also used an unsupervised learning-based technique with Austin and Irish datasets. The basic idea of this research is to find historical, and usual pattern of the power consumption and based on this each smart meter is assigned a trust score. Trust scores are used for classification and clustering. Here I will give brief information about these steps.

In order to ease mathematical tractability and use certain properties of Gaussian distribution, we convert lognormal distributions to an approximate Gaussian distribution. For this, researchers used the following equation for conversion:

$$p_t^i = \ln(P_t^i + 2) \tag{4.5}$$

After learning National Institute of Standards and Technology (NIST) recommended power transformation [5]. Where P_t^i is (t is slotted hourly) reported real power consumption with.

The proposed framework of [3] mainly consists of two parts:

- Consensus correction model.
- Trust scoring model for classification.

Consensus correction model

The main purpose of consensus correction is to avoid the consensus measure to change significantly as the result of orchestrated attacks. As a consensus measure the researchers suggested to use Ratio of Harmonic mean to Arithmetic mean instead of using Arithmetic mean or its some form of variants because arithmetic mean is not stable for legitimate changes. Let $p_t^{mix} = p_t^1, ...p_t^N$ be power consumption data series on a power transformed scale (using logarithm as shown before) gathered from N smart meters at time slot t (t is slotted hourly). The Harmonic Mean (HM) and Arithmetic Mean (AM) are defined as:

$$HM_t = \frac{N}{\sum_{i=1}^{N} \frac{1}{p_t^i}} \tag{4.6}$$

$$AM_t = \frac{\sum_{i=1}^{N} p_t^i}{N} \tag{4.7}$$

And the daily averages of these means are calculated as:

$$HM_{avg}(T) = \frac{\sum_{t=1}^{24} HM_t}{24} \tag{4.8}$$

$$AM_{avg}(T) = \frac{\sum_{t=1}^{24} AM_t}{24} \tag{4.9}$$

where T is the time window that is equal to 24 hours. Due to high fluctuations of arithmetic mean power consumption, the researchers proposed to use the ratio of $HM_{avg}(T)$ and $AM_{avg}(T)$ as the detection metric given as:

$$Q_{avg}^{ratio}(T) = \frac{HM_{avg}(T)}{AM_{avg}(T)} \tag{4.10}$$

which is much more stable than arithmetic or harmonic mean. Because of $HM \leq AM$ property, $Q_{avg}^{ratio}(T)$ cannot exceed 1 [14].

Mean and standard deviation of $Q_{avg}^{ratio}(T)$ are denoted as μ_{ration} and σ_{ratio}, respectively. Based on observations, the researchers, found out that the main properties of $Q_{avg}^{ratio}(T)$ ratio are:

- HM grows slower and decays faster than corresponding AM.
- Growth and decay rates of HM under the same δ are not the same. Decay rate is larger than growth rate for the same δ. These properties make it possible to find the type of the falsification.

Asymmetric growth and decay rates of $HM_{avg}(T)$ $AM_{avg}(T)$ cause an increase or decrease in $Q_{avg}^{ratio}(T)$. Using this, unsupervised and semi-supervised detection criteria, that show the presence of data falsification, are implemented. Therefore, there is a need to invoke a suitable consensus correction.

Let's define $Q_{avg}^{ratio}(F)$ as a sliding frame that contains cumulative average of $Q_{avg}^{ratio}(T)$ in the last F days. This average value differs from $Q_{avg}^{ratio}(F-1)$ by some threshold ϵ. Based on the properties of $Q_{avg}^{ratio}(T)$ growth/decay rates, formally, the unsupervised detection criterion is:

$Q_{avg}^{ratio}(F)$:

$\in Q_{avg}^{ratio}(F-1) \pm \sigma$ No Anomaly;

$< Q_{avg}^{ratio}(F-1) - \sigma$ Camouflage attack;

$> Q_{avg}^{ratio}(F-1) + \sigma$ Low additive attack;

where $\epsilon \in (0, 3\sigma_{ratio})$. The choice of ϵ controls whether the consensus correction step will be invoked or not [3]. With a number of experiments, the researchers concluded that $\epsilon = 2\sigma$ is the optimal choice. However, this did not work for my case. I used $\epsilon = 0.3\sigma$ as a threshold.

Once falsification is detected, observing the direction of $HM_{avg}(T)$ and $AM_{avg}(T)$ (going up or down) indicates type of falsification: additive, deductive, camouflage (Table 4.13).

Table 4.13: Inferring attack types

Ratio	**HM,AM**	**Inference**	μ_{MR}
Down	Up,Up	Additive	HM-(AM-HM)
Down	Down,Down	Deductive	AM+(AM-HM)
Down	Down,Similar	Camouflage	HM
Similar	Up,Up	Legit up	AM
Similar	Down,Down	Legit down	AM

The semi-supervised version of the detection criterion is also given in [3]. The authors of the paper gave in-detail information about how the consensus correction method works.

For simplicity from here $HM_{avg}(T)$, $AM_{avg}(T)$ are referred as HM and AM, respectively. The resilient mean ($\mu_{MR}(T)$) and standard deviation ($\sigma_{MR}(T)$) for window T are referred as μ_{MR} and σ_{MR}. The usage of ln(450) for resilient standard deviation correction is concluded empirically by the authors.

Consensus aware trust scoring model

The trust scoring model has three parts: discrete rating criterion, Folded Gaussian distribution-based weights, inverse power law kernel-based trust metric.

Discrete rating criterion. Here a discrete rating level is assigned to reported p_t^i based on its proximity to μ_{MR}. σ_{MR} is the corrected standard deviation of all p_t^i from calculated μ_{MR} in the window T. Δ_{abs} is defined as being equal to σ_{MR}. The absolute

difference between p_i^i for meter i and the μ_{MR} is denoted by $\Theta_{diff}^i = |p_t^i - \mu_{MR}|$. Based on this and Gaussian distribution, p_t^i has one of four possible rating levels.

Table 4.14: Discrete rating levels

Scenario	**Discrete Rating Level(l)**
$\Theta_{diff}^i \leq \Delta_{abs}$	4
$\Delta_{abs} < \Theta_{diff}^i \leq 2\Delta_{abs}$	3
$2\Delta_{abs} < \Theta_{diff}^i \leq 3\Delta_{abs}$	2
$Otherwise$	1

Over a time window of T hours, the ratings of each time slot t is collected and sorted as $r_{sort}^i = r_0 \leq r_1 \leq ... \leq r_{T-1}$ for each meter i.

Folded Gaussian based Weights. The normalized weights of each rating in the r_{sort}^i are denoted by $W^i = w_0, w_1, ..., w_{T-1}$. Meters with more lower rating should have lesser weights. The authors denoted $\mu_{BR} = 4$ as the highest possible rating level, and σ_{dr}^i as the standard deviation of discrete ratings of each meter with $\mu_{BR} = 4$ in a window length T.

Then a weight parameter x_t, distributed between 1 and 4, is determined as:

$$x_t = 1 + \frac{(K-1)t}{(T-1)} \qquad \forall \quad t = 0, ..., T-1 \tag{4.11}$$

where K = 4, T is window size.

And corresponding raw weight cw_t of the rating at time index t is calculated as:

$$cw_t^i = \frac{1}{\sigma_{dr}^i \sqrt{2\pi}} e^{-\frac{(x_t - \mu_{BR})^2}{2(\sigma_{dr}^i)^2}} \tag{4.12}$$

These weights are normalized by:.

$$w_t^i = \frac{cw_t^i}{\sum_{t=0}^{T-1} cw_t^i} \tag{4.13}$$

$I(l,t)$ is denoted as a function that indicates whether a particular rating level l occurs in a time slot t. Then all weights corresponding to each unique rating level $l = 1, 2, 3, 4$ within T are added and $WD(l) = \sum_{t=0}^{T-1} w_t I(l,t)$. Note that $I(l,t) = 0$ if l does not occur in time slot t and $I(l,t) = 1$ if it occurs.

The aggregate weight rating R^i of i-th meter is a continuous value in the range between 1 and 4. It is calculated as:

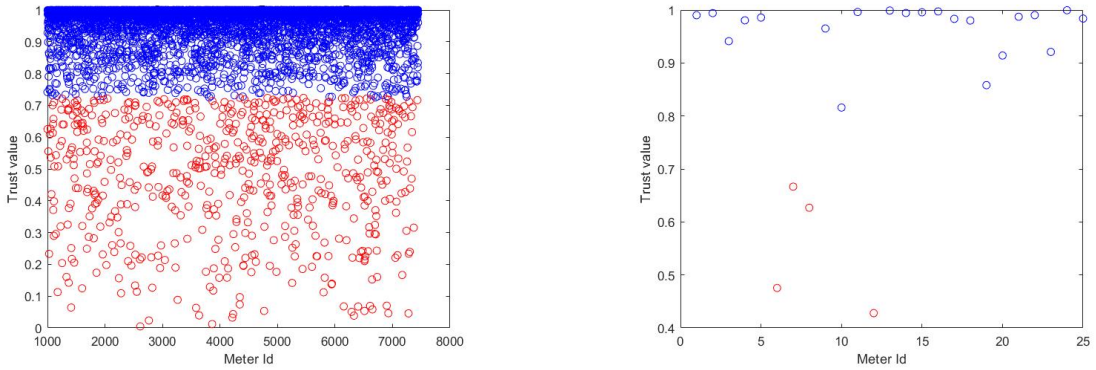
$$R^i = \sum_{l=1}^{K} l \times WD(l), \quad R^i \in \{1, 4\} \tag{4.14}$$

Inverse Power Law based Trust Value. The researchers proposed the 'inverse power law inspired kernel trick' to transform the R^i into a trust value, TR^i, as follows:

$$TR^i = \frac{1}{(K)^\eta}(R^i)^\eta, \quad TR^i \in \{0, 1\}. \tag{4.15}$$

where η is a scaling factor that controls the rate of discounting. Finally, this trust value is used for clustering.

The clustering results for Austin and Irish additive attacks are shown in Figure 4.10. For clustering, K-means algorithm is used. Results of my replication show that trust scoring based anomaly detection model did not perform as high as the authors achieved.



(a) Irish dataset (b) Austin dataset

Figure 4.10: Clustering of smart meters based on trust values

Based on the algorithms, I utilized the model proposed in [3] for smart meter datasets. The results for New York dataset are given in Table 4.15. It is clear that the highest F1-score achieved was 0.630. For additive attack type F1-score was 0.533 with precision of 0.400.

Table 4.15: The results of Bhattacharjee et al's research work replication on New York dataset

	Precision	**Recall**	**F1-score**
Additive	0.800	0.400	0.533
Deductive	0.833	0.507	0.630
Camouflage	0.833	0.507	0.630

Low performance can also bee seen in Tables 4.16, 4.17 where Austin and Irish datasets have been examined. Based on the main results of this thesis, given in 4.4,

the proposed model of the thesis without sliding window outperforms the replicated trust scoring model given in [3] on the smart meters datasets.

However, this might be the result of the lack of information or parameters given in the paper because some parameters and values were chosen based on the experiments they conducted and they were not defined fully. The performance they actually achieved were False-positive rate of 13%, Missed detection rate of 9%.

Table 4.16: The results of Bhattacharjee et al's research work replication on Austin dataset

	Precision	**Recall**	**F1-score**
Additive	0.300	0.750	0.429
Deductive	0.300	0.750	0.429
Camouflage	0.500	0.714	0.588

Table 4.17: The results of Bhattacharjee et al's research work replication on Irish (25) dataset

	Precision	**Recall**	**F1-score**
Additive	0.60	0.666	0.632
Deductive	0.50	0.833	0.625
Camouflage	0.50	0.550	0.526

Given trust scoring models, I also used two unsupervised learning algorithms, namely, Isolation Forest (IF) and Lightweight On-line Detector of Anomalies (LODA) on additive attack type of New York, Austin, Irish datasets to detect compromised smart meters. The purpose of this experiment was to compare IF and LODA with K-means algorithm I used above. These comparison results can be found in C.

4.9 Summary

This chapter has covered parameters that are used for Hierarchical Self-Organizing Maps and for datasets along with attack injection. Then performance metrics that are used for measuring the detection rate and performance of the proposed model have been given. How SOM maps and neurons approximate the input data on the first layer lattice has also been demonstrated. The distance matrices have made it clear that upper layer maps are needed because the data instances of different classes have not been separated in the first layer SOM.

Main experimental results demonstrate that the proposed model could achieve F1-score of as high as 0.980 with a precision of 0.984 (New York deductive dataset, Table 4.3). Then the model is tested on another IoT dataset, the Power System Attacks dataset where no less than 0.972 of precision, recall, F1 scores have been achieved. As another set of experiments, a sliding window approach has been used with and without window overlaps. The results seem to indicate that further temporal information is not required when there is explicit time information in the form of a timestamp for Smart Meter datasets. However, this observation might need to be confirmed with further experiments in the future. Examination of unseen behaviors of input data has provided deeper analysis on the upper layer SOM maps. With the help of the map partitioning feature of SOM, clustering of data on a 2D map has been explored. Besides, the number of clusters for all the datasets is studied. Also, I have compared the results of the model of this thesis with the results of the approach given by Bhattacharjee at al. [32] by replicating their work.

Chapter 5

Conclusion

The number of IoT devices has been increasing rapidly and is expected to grow even further in the coming years. This expansion comes with the challenge of creating reliable defense mechanisms against both existing attacks and new ones to be introduced by adversaries. The defense systems need to have certain properties such as robustness, scalability, and generalization. Considering this, the main objective of this thesis was to analyze how effective a Hierarchical SOM model could be as an anomaly detector for various IoT networks.

With the use of different datasets, it is shown that the model could achieve precision between 0.887 and 0.984, recall between 0.829 and 0.982, F1-scores between 0.857 and 0.980. Testing various attack types across all smart meter datasets has shown how robust the model is for different data falsification attacks. For the Irish dataset, different numbers of smart meters have also been tested and the scalability property of the model has been confirmed.

Additionally, the use of a sliding window approach with different window overlaps has been explored. The results showed that even with only three features, a Hierarchical SOM model is able to achieve a high F1-score for additive, deductive, and camouflage attacks without using a sliding window. This indicates that using explicit timestamps might eliminate the need for further temporal information in the form of sliding windows. The proposed model can also achieve high performance for a power framework type of IoT systems without the use of a sliding window approach.

Moreover, the proposed model that is shown to be robust on smart meter datasets is also employed on high dimensional Power System Attacks dataset with the same properties (in terms of sliding window and window overlap selections). And the performance has been at least 0.972, showing the generalization property of the model. Therefore, the model can be used in IoT systems of different devices.

Furthermore, other advantages of Self-Organizing Maps such as visualization of

high-dimensional data on two or three-dimensional graphs and clustering have also been analyzed. While the distance matrix gives information about clustering, data distribution on neurons demonstrates how SOM nodes are approximating the input data. The option of selecting a map size and other properties can present flexibility so that based on the nature of IoT networks, a model can be adjusted easily. Upper layer maps with smaller sizes make it possible to implement parallel computing and therefore reduce the run-time in real-life applications.

5.1 Future work

Future work will investigate the performance of the proposed Hierarchical SOM model on more datasets and attacks in order to further examine its performance. Given the absence of the smart meter datasets containing attacks, attack injection needed to be simulated. Thus, one goal is to obtain real datasets which are captured under attack conditions and use them for the evaluations. Additionally, the proposed model can be further generalized so that it can work with not only datasets with binary labels, but the ones with multi-class labels. Finally, further analysis into camouflage attacks for smart meters could improve the performance of the proposed model on these types of behaviors. A deeper examination and a revised approach for camouflage attacks might be able to improve the performance even further.

Bibliography

[1] Armin Aligholian, Mohammad Farajollahi, and Hamed Mohsenian-Rad. Unsupervised learning for online abnormality detection in smart meter data. In *2019 IEEE Power Energy Society General Meeting (PESGM)*, pages 1–5, August 2019.

[2] Eduardo Werley S. Angelos, Osvaldo R. Saavedra, Omar A. Carmona Cortés, and André Nunes de Souza. Detection and identification of abnormalities in customer consumptions in power distribution systems. *IEEE Transactions on Power Delivery*, 26(4):2436–2442, October 2011.

[3] Shameek Bhattacharjee, Aditya Thakur, and Sajal K. Das. Towards fast and semi-supervised identification of smart meters launching data falsification attacks. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18, page 173–185, New York, NY, USA, 2018. Association for Computing Machinery.

[4] Raymond C. Borges Hink, Justin M. Beaver, Mark A. Buckner, Tommy Morris, Uttam Adhikari, and Shengyi Pan. Machine learning for power system disturbance and cyber-attack discrimination. In *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, pages 1–8, August 2014.

[5] G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252, 1964.

[6] Efrem Heri Budiarto, Adhistya Erna Permanasari, and Silmi Fauziati. Unsupervised anomaly detection using k-means, local outlier factor and one class svm. In *2019 5th International Conference on Science and Technology (ICST)*, volume 1, pages 1–5, July 2019.

[7] Cisco Systems, Inc. Cisco Annual Internet report. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, March 2020. Accessed: 2021-12-14.

[8] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, April 1979.

[9] Meng Fanlin and Ye Wei. Summary of research on security and privacy of smart grid. In *2020 International Conference on Computer Communication and Network Security (CCNS)*, pages 39–42, August 2020.

[10] Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain. Machine learning in iot security: Current solutions and future challenges. *IEEE Communications Surveys Tutorials*, 22(3):1686–1721, April 2020.

[11] Irish Social Science Data Archive (ISSDA). CER Smart Metering Project. https://www.ucd.ie/issda. Accessed: 2021-12-15.

[12] J. P. Esa Alhoniemi, Johan Himberg and J. Vesanto. SOM Toolbox. http://www.cis.hut.fi/somtoolbox/. Accessed: 2021-12-14.

[13] Rong Jiang, Rongxing Lu, Ye Wang, Jun Luo, Changxiang Shen, and Xuemin Shen. Energy-theft detection issues for advanced metering infrastructure in smart grid. *Tsinghua Science and Technology*, 19(2):105–120, April 2014.

[14] Anish Jindal, Amit Dua, Kuljeet Kaur, Mukesh Singh, Neeraj Kumar, and S. Mishra. Decision tree and svm-based data analytics for theft detection in smart grid. *IEEE Transactions on Industrial Informatics*, 12(3):1005–1016, June 2016.

[15] Paria Jokar, Nasim Arianpoo, and Victor C. M. Leung. Electricity theft detection in ami using customers' consumption patterns. *IEEE Transactions on Smart Grid*, 7(1):216–226, 2016.

[16] 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, 2003.*, volume 3, pages 1808–1813 vol.3, July 2003.

[17] R. Kochendörffer. Kreyszig, e.: Advanced engineering mathematics. j. wiley & sons, inc., new york, london 1962. ix + 856 s. 402 abb. preis s. 79.—. *Biometrische Zeitschrift*, 7(2):129–130, 1965.

[18] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990.

[19] Teuvo Kohonen. *MATLAB Implementations and Applications of the Self-Organizing Map*. Unigrafia Oy, Helsinki, Finland, 2014.

[20] Teuvo Kohonen and Manfred Schroeder. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, January 2001.

[21] Varun Badrinath Krishna, Kiryung Lee, Gabriel A. Weaver, Ravishankar K. Iyer, and William H. Sanders. F-deta: A framework for detecting electricity theft attacks in smart grids. In *Proceedings - 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016*, pages 407–418, United States, September 2016. Institute of Electrical and Electronics Engineers Inc.

[22] Duc C. Le. An unsupervised learning approach for network and system analysis. Msc thesis, Dalhousie University, Halifax, Nova Scotia, Canada, 2017.

[23] Duc C. Le, A. Nur Zincir-Heywood, and Malcolm I. Heywood. Data analytics on network traffic flows for botnet behaviour detection. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, December 2016.

[24] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[25] J. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

[26] Pascal Maniriho, Ephrem Niyigaba, Zephanie Bizimana, Valens Twiringiyimana, Leki Jovial Mahoro, and Tohari Ahmad. Anomaly-based intrusion detection approach for iot networks using machine learning. In *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, pages 303–308, November 2020.

[27] Daisuke Mashima and Alvaro A. Cárdenas. Evaluating electricity theft detectors in smart grid networks. In Davide Balzarotti, Salvatore J. Stolfo, and Marco Cova, editors, *Research in Attacks, Intrusions, and Defenses*, pages 210–229, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[28] McLaughlin, Stephen and Podkuiko, Dmitry and McDaniel, Patrick. Energy theft in the advanced metering infrastructure. In Erich Rome and Robin Bloomfield, editors, *Critical Information Infrastructures Security*, pages 176–187, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[29] Andrew Meola. How the Internet of Things and smart meters are streamlining gas, water, and electric data transmissions. https://www.internetsociety.org/resources/doc/2015/iot-overview/, January 2020. Accessed: 2021-12-15.

[30] Dubravko Miljković. Brief review of self-organizing maps. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1061–1066, May 2017.

[31] Hadeel S. Obaid, Saad Ahmed Dheyab, and Sana Sabah Sabry. The impact of data pre-processing techniques and dimensionality reduction on the accuracy of machine learning. In *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)*, pages 279–283, March 2019.

[32] Pecan Street Inc. Dataport. Pecan Street Dataport. Accessed: 2021-12-15.

[33] Ravi Ponmalai and Chandrika Kamath. Self-organizing maps and their applications to data analysis. *U.S. Department of Energy Office of Scientific and Technical Information*, September 2019.

[34] Karen Rose, Scott D. Eldridge, and Lyman Chapin. THE INTERNET OF THINGS : AN OVERVIEW Understanding the Issues and Challenges of a More Connected World. https://www.internetsociety.org/resources/doc/2015/iot-overview/, October 2015. Accessed: 2021-12-15.

[35] Asma Saleem, Khadim Hussain Asif, Ahmad Ali, Shahid Mahmood Awan, and Mohammed A. Alghamdi. Pre-processing methods of data mining. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 451–456, December 2014.

[36] Iqbal H. Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):160, March 2021.

[37] J. Sola and J. Sevilla. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*, 44(3):1464–1468, June 1997.

[38] Hugo Steinhaus. Sur la division des corps matériels en parties. *Bull. Acad. Pol. Sci., Cl. III*, 4:801–804, 1957.

[39] The MathWorks, Inc. datenum. https://www.mathworks.com/help/matlab/ref/datenum.html. Accessed: 2021-12-15.

[40] Tommy Morris. Industrial Control System (ICS) Cyber Attack Datasets. https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets. Accessed: 2021-12-15.

[41] Muhammadjon Toshpulatov and Nur Zincir-Heywood. Anomaly Detection on Smart Meters using Hierarchical Self Organizing Maps. In *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6, September 2021.

[42] Imtiaz Ullah and Qusay H. Mahmoud. A scheme for generating a dataset for anomalous activity detection in iot networks. In Cyril Goutte and Xiaodan Zhu, editors, *Advances in Artificial Intelligence*, pages 508–520, Cham, 2020. Springer International Publishing.

[43] Uttam Adhikari, Shengyi Pan, Tommy Morris and Raymond Borges, Justin Beaver. Power system. https://www.kaggle.com/bachirbarika/power-system. Retrieved: 2021-01-14.

[44] J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586–600, May 2000.

[45] Jonathan White and Phil Legg. Unsupervised one-class learning for anomaly detection on home iot network devices. In *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pages 1–8, June 2021.

[46] Wikipedia contributors. Self-organizing maps, 2011. [Online; Accessed: 2021-12-15].

[47] Xiaofang Xia, Yang Xiao, Wei Liang, and Meng Zheng. Gthi: A heuristic algorithm to detect malicious users in smart grids. *IEEE Transactions on Network Science and Engineering*, 7(2):805–816, April 2020.

[48] Wei Yu, David Griffith, Linqiang Ge, Sulabh Bhattarai, and Nada Golmie. An integrated detection system against false data injection attacks in the smart grid. *Sec. and Commun. Netw.*, 8(2):91–109, January 2015.

Appendix A

Type of scenarios in Power Systems Attacks dataset

Type of scenarios:

1. **Short-circuit fault** - this is a short in a power line and can occur in various locations along the line, the location is indicated by the percentage range.
2. **Line maintenance** - one or more relays are disabled on a specific line to do maintenance for that line
3. **Remote tripping command injection (Attack)** - this is an attack that sends a command to a relay that causes a breaker to open. It can only be done once an attacker has penetrated outside defenses.
4. **Relay setting change (Attack)** - relays are configured with a distance protection scheme and the attacker changes the setting to disable the relay function such that the relay will not trip for a valid fault or a valid command.
5. **Data Injection (Attack)** - here a valid fault is imitated by changing values to parameters such as current, voltage, sequence components, etc. This attack aims to blind the operator and causes a blackout.

Table A.1: Natural event scenarios

Scenario	**Natural events type**
	SLG Faults
1	Fault from 10-19% on L1
2	Fault from 20-79% on L1
3	Fault from 80-90% on L1
4	Fault from 10-19% on L2
5	Fault from 20-79% on L2
6	Fault from 80-90% on L2
	Line maintenance
13	Line L1 maintenance
14	Line L2 maintenance

Table A.2: Attack event scenario

Scenario	**Attack type**
	Data injection
7	Fault from 10-19% on L1 with tripping command
8	Fault from 20-79% on L1 with tripping command
9	Fault from 80-90% on L1 with tripping command
10	Fault from 10-19% on L2 with tripping command
11	Fault from 20-79% on L2 with tripping command
12	Fault from 80-90% on L2 with tripping command
	Remote Tripping Command Injection
	Sub-type - Command injection against single relay
15	Command Injection to R1
16	Command Injection to R2
17	Command Injection to R3
18	Command Injection to R4
	Sub-type - Command injection against two relays
19	Command Injection to R1 and R2
20	Command Injection to R3 and R4
	Relay Setting Change
	Disabling relay function - single relay disabled & fault
21	Fault from 10-19% on L1 with R1 disabled & fault
22	Fault from 20-90% on L1 with R1 disabled & fault
23	Fault from 10-49% on L1 with R2 disabled & fault
24	Fault from 50-79% on L1 with R2 disabled & fault
25	Fault from 80-90% on L1 with R2 disabled & fault
26	Fault from 10-19% on L2 with R3 disabled & fault
27	Fault from 20-49% on L2 with R3 disabled & fault
28	Fault from 50-90% on L2 with R3 disabled & fault
29	Fault from 10-79% on L2 with R4 disabled & fault
30	Fault from 80-90% on L2 with R4 disabled & fault
	Disabling relay function - two relays disabled & fault
35	Fault from 10-49% on L1 with R1 and R2 disabled & fault
36	Fault from 50-90% on L1 with R1 and R2 disabled & fault
37	Fault from 10-49% on L1 with R3 and R4 disabled & fault
38	Fault from 50-90% on L1 with R3 and R4 disabled & fault
	Disabling relay function - two relay disabled & line maintenance
39	L1 maintenance with R1 and R2 disabled
40	L1 maintenance with R1 and R2 disabled

Table A.3: No event scenario

Scenario	**No Events (Normal operation)**
41	Normal Operation load changes

Table A.4: Classification of events

	Natural events	**Attack events**	**No events**
Scenarios	1, 2, 3, 4, 5, 6, 13, 14	7, 8, 9, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 35, 36, 37, 38, 39, 40	41

Type of events are given in Table A.1, A.2, A.3 and their classification is given in Table A.4.

The 128 features are explained here. Phasor measurement units (PMU) provide 29 types of measurements. In the Power System framework, there are four PMUs giving 116 measurements, or features, altogether. The index of each column is in the form of "R#-Signal reference" indicating PMU specified by "R#". Signal references along with their description are given in the table Table A.5. For example, R3-PAA:VH means Phase A voltage phase angle measured by PMU R3. Apart from PMU measurements, there are 12 columns for control panel logs, Snort alerts, and relay logs of 4 PMU/relay as relay and PMU are integrated together.

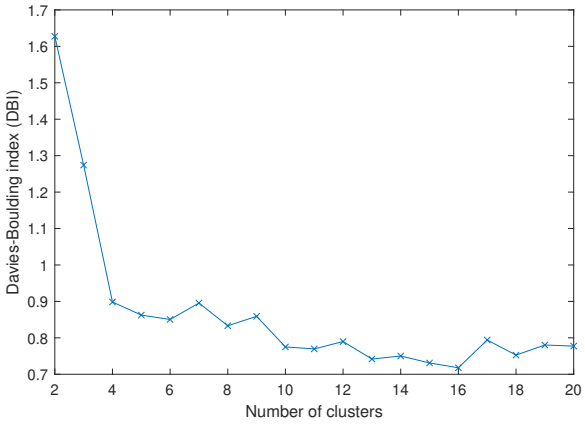
Table A.5: Power System Attacks dataset features

Feature	**Description**
PA1:VH – PA3:VH	Phase A - C Voltage Phase Angle
PM1: V – PM3: V	Phase A - C Voltage Phase Magnitude
PA4:IH – PA6:IH	Phase A - C Current Phase Angle
PM4: I – PM6: I	Phase A - C Current Phase Magnitude
PA7:VH – PA9:VH	Pos. – Neg. – Zero Voltage Phase Angle
PM7: V – PM9: V	Pos. – Neg. – Zero Voltage Phase Magnitude
PA10:VH - PA12:VH	Pos. – Neg. – Zero Current Phase Angle
PM10: V - PM12: V	Pos. – Neg. – Zero Current Phase Magnitude
F	Frequency for relays
DF	Frequency Delta (dF/dt) for relays
PA:Z	Appearance Impedance for relays
PA:ZH	Appearance Impedance Angle for relays
S	Status Flag for relays

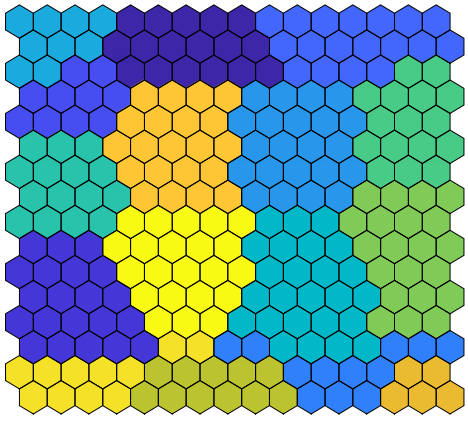
Appendix B

Clustering visualizations

Following is a full list of clustering visualizations and the Davies-Boulding Index associated with each clustering for all datasets.

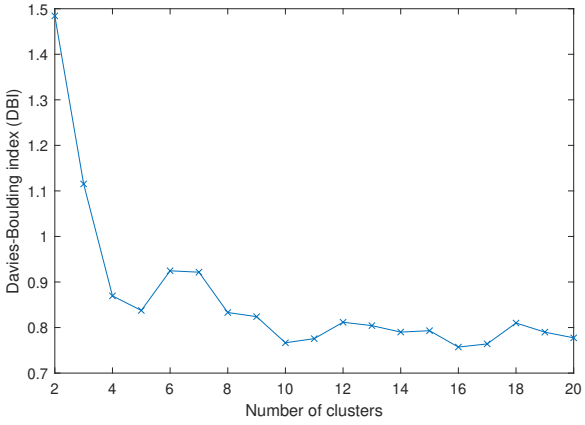


(a) Davies-Boulding clustering index

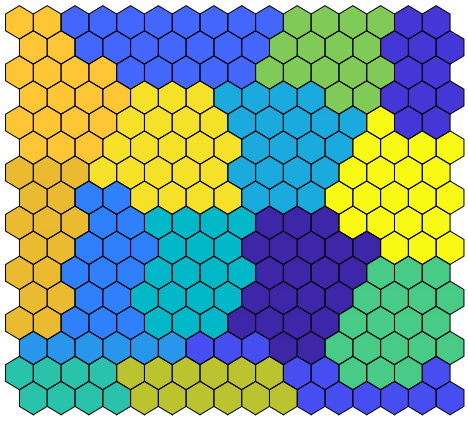


(b) Map partitioning

Figure B.1: Clusters formed on the trained SOM: New York Additive

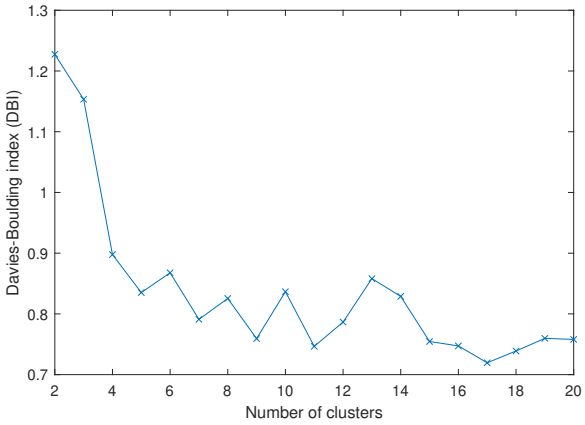


(a) Davies-Boulding clustering index

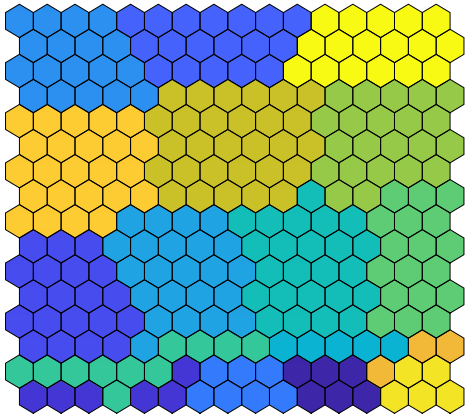


(b) Map partitioning

Figure B.2: Clusters formed on the trained SOM: New York Deductive

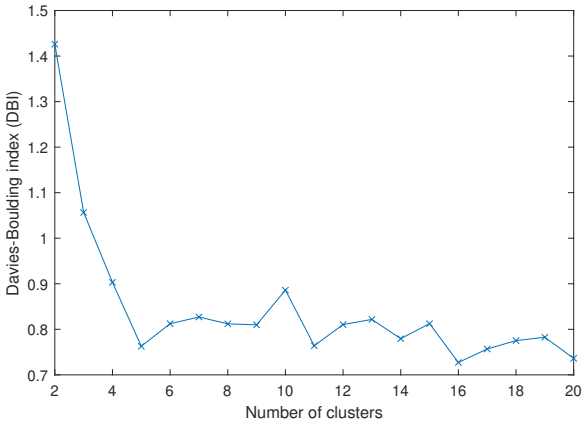


(a) Davies-Boulding clustering index

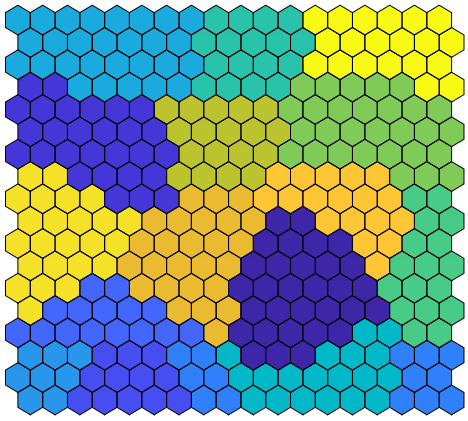


(b) Map partitioning

Figure B.3: Clusters formed on the trained SOM: New York Camouflage

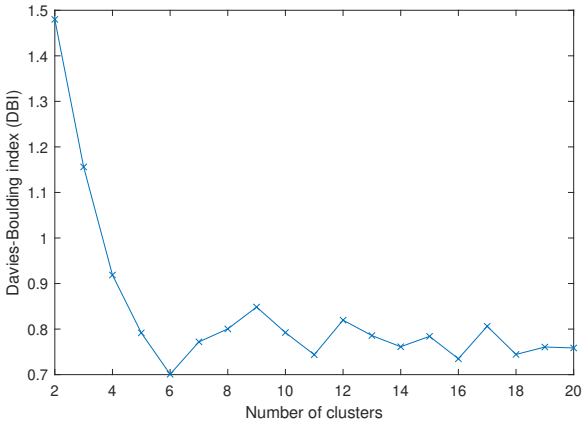


(a) Davies-Boulding clustering index

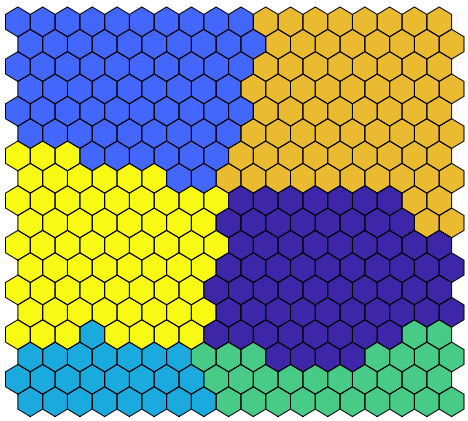


(b) Map partitioning

Figure B.4: Clusters formed on the trained SOM: Austin Additive

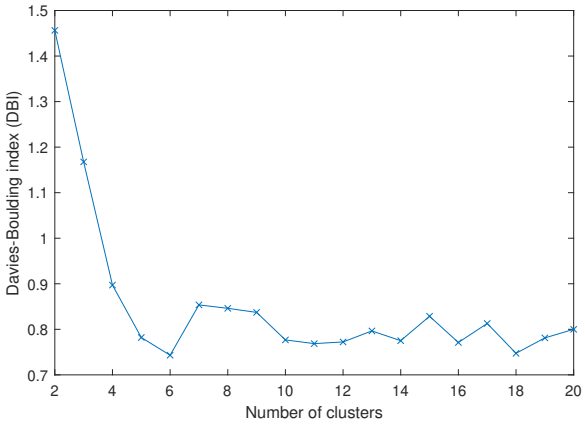


(a) Davies-Boulding clustering index

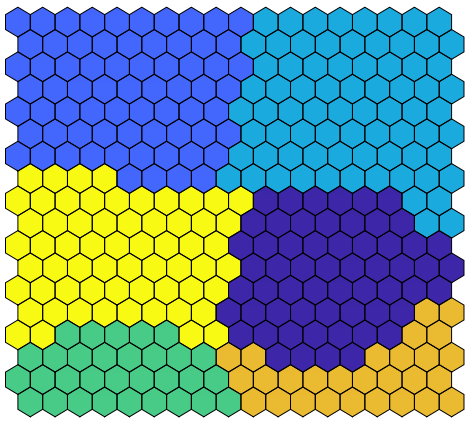


(b) Map partitioning

Figure B.5: Clusters formed on the trained SOM: Austin Deductive

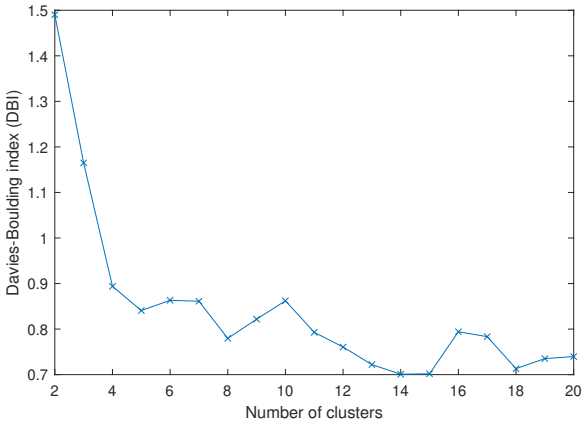


(a) Davies-Boulding clustering index

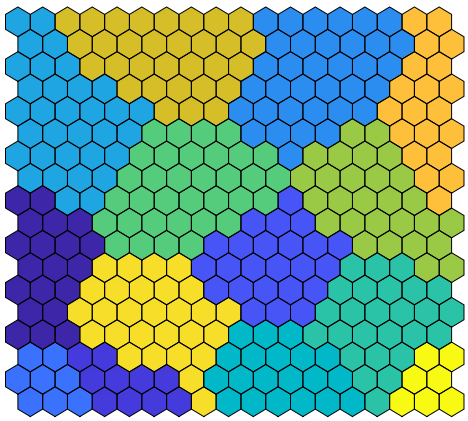


(b) Map partitioning

Figure B.6: Clusters formed on the trained SOM: Austin Camouflage

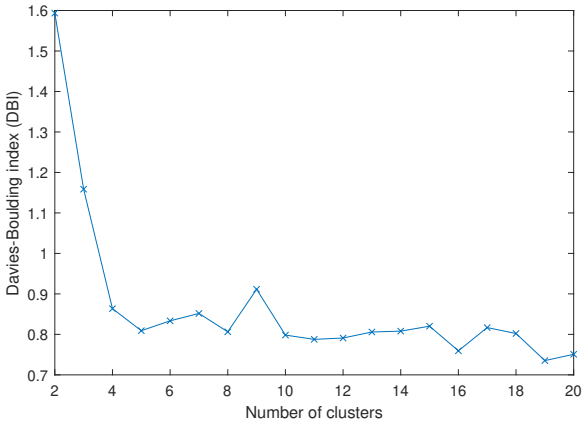


(a) Davies-Boulding clustering index

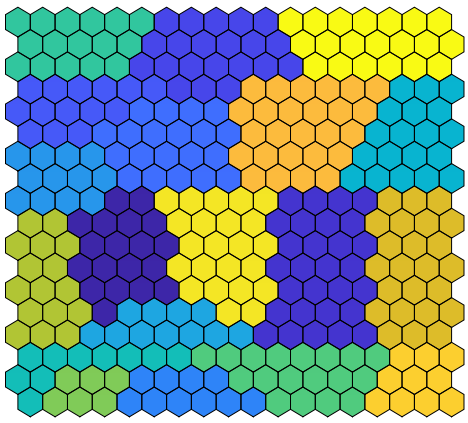


(b) Map partitioning

Figure B.7: Clusters formed on the trained SOM: Irish 25 Additive

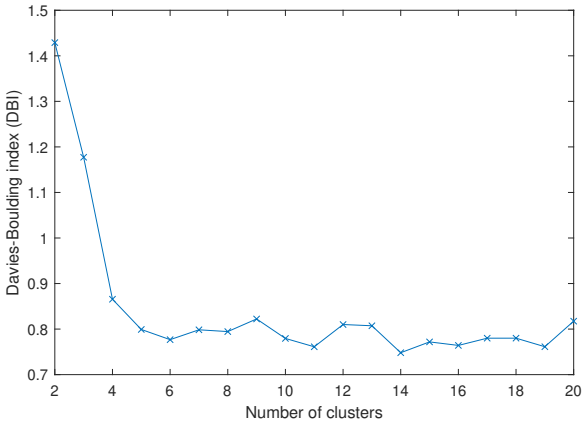


(a) Davies-Boulding clustering index

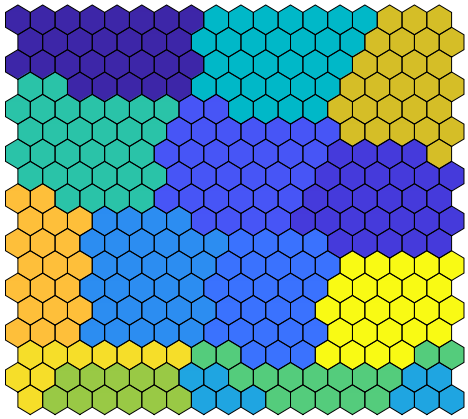


(b) Map partitioning

Figure B.8: Clusters formed on the trained SOM: Irish 25 Deductive

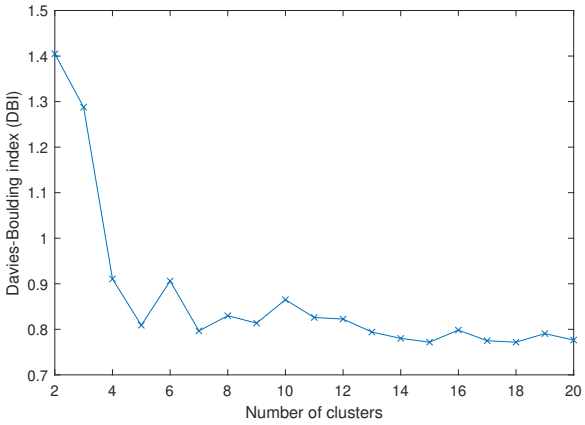


(a) Davies-Boulding clustering index

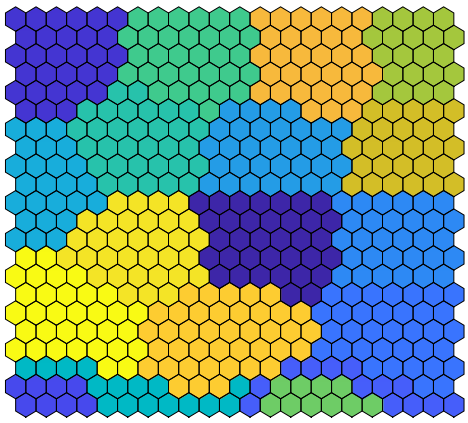


(b) Map partitioning

Figure B.9: Clusters formed on the trained SOM: Irish 25 Camouflage

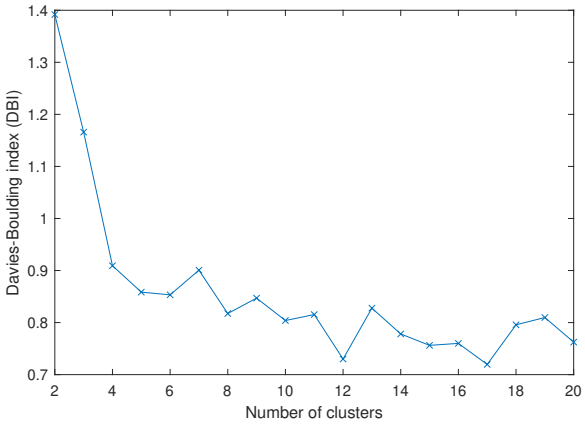


(a) Davies-Boulding clustering index

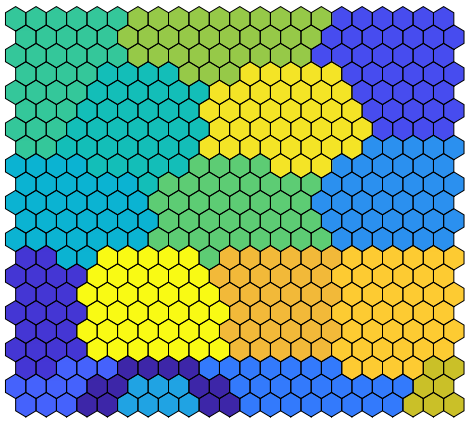


(b) Map partitioning

Figure B.10: Clusters formed on the trained SOM: Irish 100 Additive

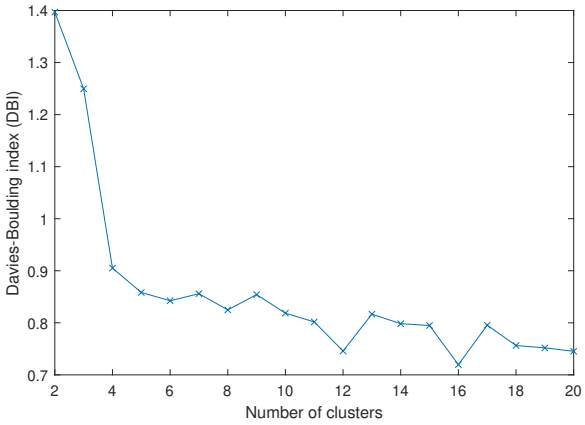


(a) Davies-Boulding clustering index

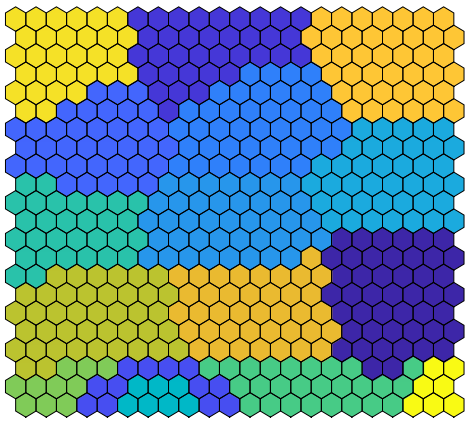


(b) Map partitioning

Figure B.11: Clusters formed on the trained SOM: Irish 100 Deductive

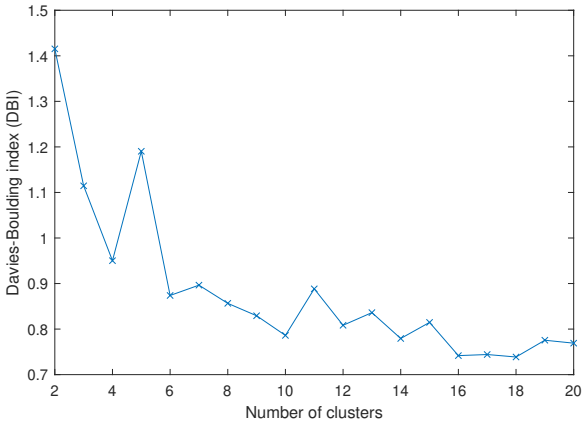


(a) Davies-Boulding clustering index

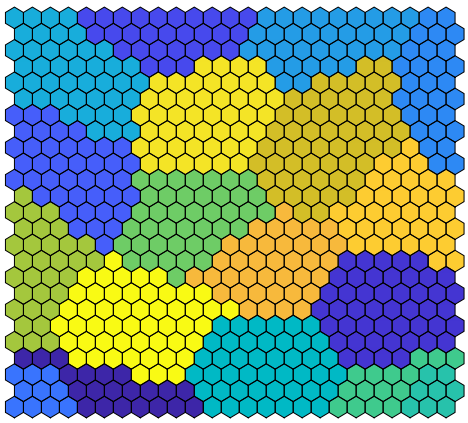


(b) Map partitioning

Figure B.12: Clusters formed on the trained SOM: Irish 100 Camouflage

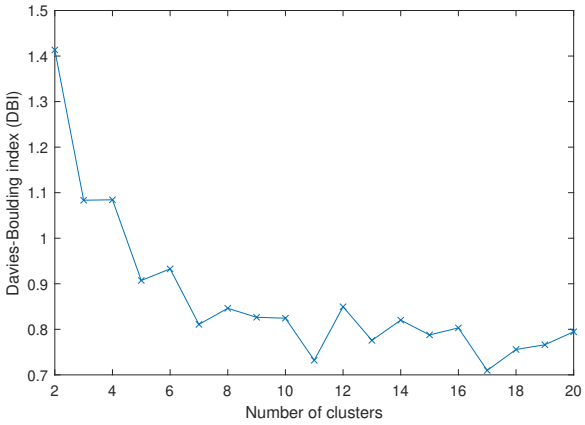


(a) Davies-Boulding clustering index

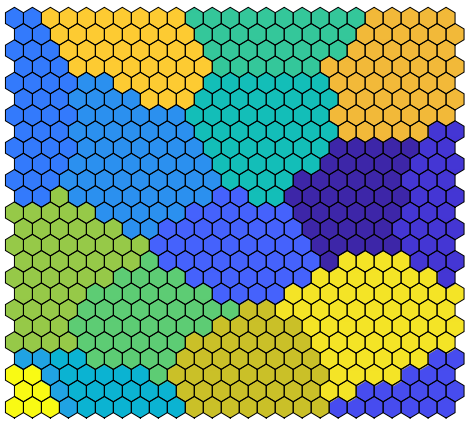


(b) Map partitioning

Figure B.13: Clusters formed on the trained SOM: Irish 200 Additive

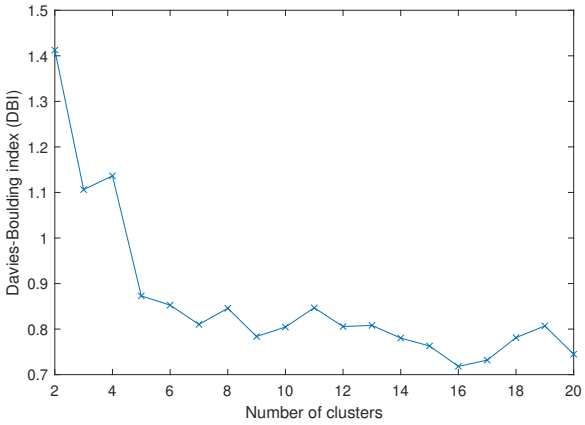


(a) Davies-Boulding clustering index

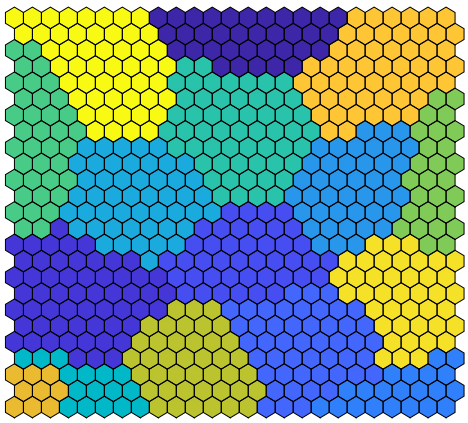


(b) Map partitioning

Figure B.14: Clusters formed on the trained SOM: Irish 200 Deductive

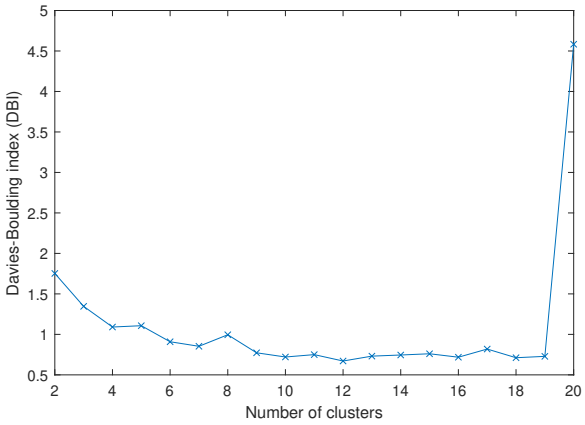


(a) Davies-Boulding clustering index

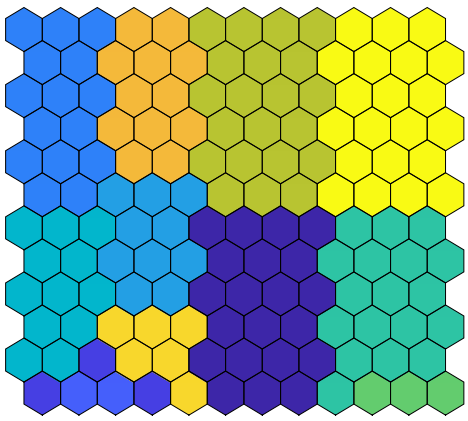


(b) Map partitioning

Figure B.15: Clusters formed on the trained SOM: Irish 200 Camouflage

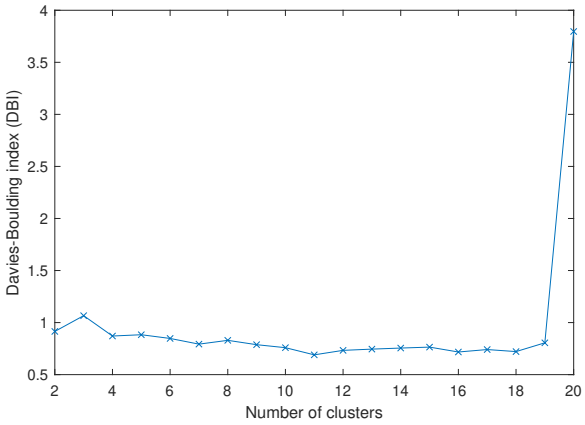


(a) Davies-Boulding clustering index

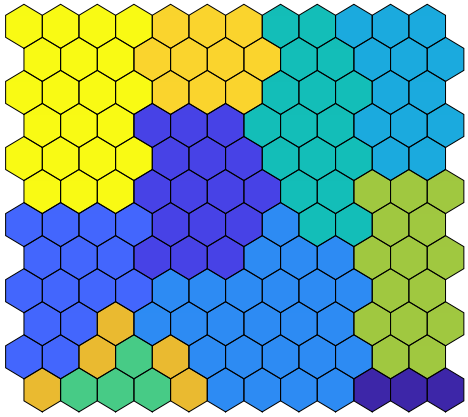


(b) Map partitioning

Figure B.16: Clusters formed on the trained SOM: PSA Full

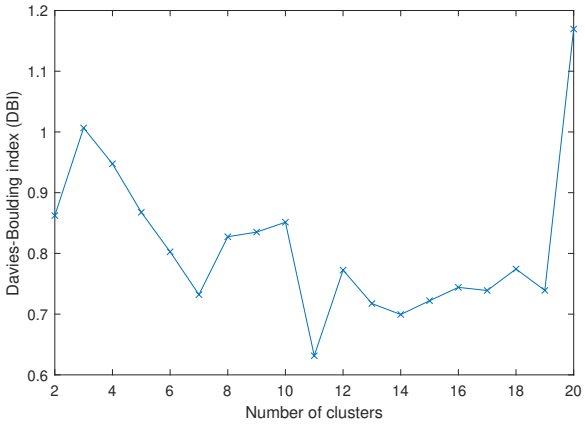


(a) Davies-Boulding clustering index

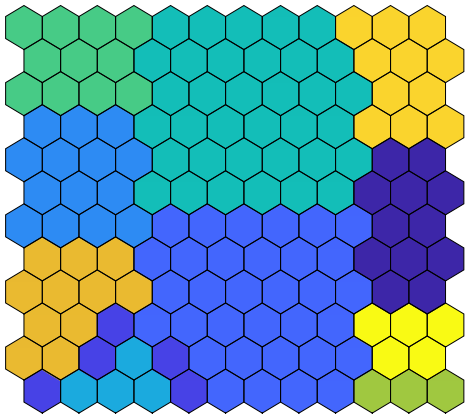


(b) Map partitioning

Figure B.17: Clusters formed on the trained SOM: PSA One PMU with Extra features



(a) Davies-Boulding clustering index



(b) Map partitioning

Figure B.18: Clusters formed on the trained SOM: PSA One PMU only

Appendix C

Comparison research

The following tables are the results of two unsupervised learning approaches, namely Isolation Forest, and Lightweight On-line Detector of Anomalies on New York, Austin, Irish dataset with additive data falsification. Note that all smart meters have trust values assigned based on [32].

Table C.1: Isolation Forest on Smart meters with trust values

	Precision	**Recall**	**F1-score**
New York	0.500	0.556	0.526
Austin	0.300	0.273	0.286
Irish	0.374	0.667	0.480

Table C.2: Lightweight On-line Detector of Anomalies on Smart meters with trust values

	Precision	**Recall**	**F1-score**
New York	0.200	1.00	0.333
Austin	0.200	0.667	0.308
Irish	0.085	0.297	0.132