**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**Network Intrusion Detection via
Distributed Machine Learning On
Smart Gateway Network**

Suleman Khalid Rai

**School of Electrical & Electronic
Engineering**

A thesis submitted to the Nanyang Technological
University in partial fulfillment of the requirement for
the degree of Master of Engineering

March 2017

Acknowledgement

During my time spent at NTU for my master of engineering program, i have come across several individuals who have helped me in my research work. As such i am greatly thankful to these people for their help and insight. With their guidance and help i was able to make most of this great learning opportunity and complete my research.

First and foremost, i am eternally grateful and gratuitous towards my supervisor Professor Yu Hao in providing me an opportunity to pursue my graduate studies under his supervision. His feedback, advice and support enabled me for a successful completion of my studies. Professor Yu Hao always encouraged me to work hard and be enthusiastic towards research. Under his supervision i have learned new skills which will benefit me in future.

Secondly i am grateful towards the confirmation exercise examiners Professor Soh Yeng Chai, Professor Lipo Wang and Professor Tay Wee Peng for accessing my project. I will also like to thank the staff of graduate programme office for their assistance in administrative matters for my graduate studies. I am also grateful to JTC Corporation and Energy Research Institute @ NTU in providing me an opportunity to work on their projects and would like to express special thanks to Dr. Koh Leong Hai, Mr. Jatin Narotam Sarvaiya, Mr. Krishnamoorthy Baskaran, Dr. Swapnil Dubey and Dr. Dominic Frank Maurath, Mdm Chan Kam Sim Sally, Mdm Nurhidayah Amaradasa Meli and Mdm Lily Lim Seok Kim from ERIAN.

Thirdly i am grateful to the VIRTUS faculty members, Professor Chang Joseph Sylvester, Professor Chan Pak Kwong, Professor Jong Ching Chuen and Professor Siek Liter for providing me with an opportunity to work in VIRTUS. I also want to thank the VIRTUS and EEE GPO staff Mr. David

Robert Neubronner, Mrs. Lim-Tan Gek Eng, Mr. Sia Liang Poo and others who have helped in administrative matters. I am specially grateful to my colleagues at VIRTUS, Huang Hantao, Cai Yuehua, Xu Hang, Sai Manoj Pudukotai Dinakarrao, Wang Kanwen, Xu Dongjun, Li Nan, Shang Yang, Ma shunli, Wang Yuhao, Huang Xiwei, Li Peng, Shen Shanlan, Guo Jing, Chen Shuai, Wang Xiaolong, Liu Xu, Yang Chang, Jiang Yu and many others with whom i worked very closely and developed good friendship which made my graduate studies a memorable experience. I am also thankful to Zhang Yonghao who developed the home automation system and the associated android application and was always willing to help me with technical details.

Lastly i am thankful to my family and friends, especially my parents, brother and my spouse for sacrificing my presence and their happiness so that i am successful and happy in my career goals. Their encouragement, patience and support has enabled me to remain motivated in completing my graduate studies.

Abstract

The future smart building requires the deployment of smart grid to not only provide physical power supply but to also make cyber control with energy savings. A smart home is a typical cyber-physical system with linkage between physical power grid and cyber information network. Usually the linkage here can be a data server or a gateway with collected personal data of energy, temperature, humidity, Wireless Fidelity (Wi-Fi) etc. This however may result in a serious security concern if the data server or gateway are intruded with malicious attacks. Vulnerabilities inherent to communication and computing of a cyber-physical system such as smart home can affect the entire infrastructure. Various methods have been proposed to detect these intrusions and prevent them from causing harm to the smart home system. This thesis studies machine learning approach to identify malicious intrusions and explores the performance improvement in time complexity and classification accuracy through distributed machine learning algorithms.

In this research, supervised machine learning algorithms which are studied are Support Vector Machine (SVM) and Extreme Learning Machine (ELM). Two publicly available datasets NSL-KDD and ISCX 2012 are evaluated to compare the performance of SVM and ELM. Experiments are performed for both binary and multi class classification to evaluate the performance of these classifiers on the benchmark datasets. For SVM, four different kernel functions are used for binary and multi class classification. For NSL-KDD benchmark, the sigmoid kernel function gives the best classification accuracy and has reasonable time complexity when compared against other kernel functions. For ISCX 2012 dataset, Linear and Sigmoid Kernel functions give comparable performance metrics. Experiment results also show that classification performance of SVM is superior to ELM for most

of the intrusion classes. ELM, on the other hand, offers a better performance in time complexity. To further improve the time complexity of ELM, an incremental least square matrix solver based on Cholesky decomposition is utilized. Because of the data imbalance, ELM does not give good performance metrics on the Remote to Local (R2L) and User to Root (U2R) classes. To resolve this data imbalance, weighted ELM is proposed to improve the classification performance for these classes. Experiment results show that using Weighted ELM, the sensitivity improves from 0 % to 77.6 % for the U2R class and from 0 % to 32.9 % for the Remote to Local (R2L) class in comparison to the basic ELM. The precision improves slightly from 0 % to 2 % for the User to Root (U2R) class and improves from 23.3 % to 46.8 % for the Remote to Local (R2L) class.

Conventional machine learning approaches usually follow a centralized data analytic architecture with all the computation performed on a single central server. In the off line training phase depending upon the amount of data to be processed, this can be a huge workload specially on the resource limited smart gateways. To overcome this issue, a distributed machine learning algorithm Sequential Distributed SVM (SQ-DSVM) is proposed to resolve the off line training time complexity for intrusion detection. Experimental results show that with 5 smart gateways working in parallel, the proposed distributed algorithm Sequential Distributed SVM (SQ-DSVM) can accomplish a performance improvement in run time up to $5\times$ to centralized SVM. Apart from Distributed Support Vector Machine (DSVM), computationally efficient Distributed Neuron Network (DNN) for intrusion detection is also considered. Experiment results show that for NSL-KDD dataset using a Distributed Neuron Network (DNN) with 5 subsystems, the server processing time improves by a factor of $12\times$ when compared to SVM and up to $5\times$ when compared to centralized network intrusion detection system (NIDS) using only a single neural network. For ISCX dataset server processing time improves by a factor of $20\times$ when compared to SVM and by a factor of $5\times$ when compared to centralized NIDS using only a single neural network. In real time, intrusion detection latency is important to detect intrusions. Queuing delay analysis is done using an M/M/1 queuing model. Experiment results show that using Distributed Neuron Network (DNN),

the queuing delay and total delay are reduced proportionally to the number of sub systems used in a Distributed Neuron Network (DNN). To improve the classification performance for different classes, a hybrid classification scheme is used which combines SVM and ELM into a single classification system. Experiment results show SVM should be used for detecting Denial of Service (DOS) and User to Root (U2R) classes whereas ELM should be used for detecting Probe and Remote to Local (R2L) classes.

In practice labeling the data for normal and intrusion instances is quite an expensive human operation. To overcome this issue an unsupervised machine learning algorithm, Unsupervised Extreme Learning Machine (US-ELM) is used to classify instances into 2 different clusters for normal and intrusive instances. Experiment results show that clustering accuracy increases with the embedding dimension and achieves a peak clustering accuracy of 86.9 % with the embedding dimension of 12. Using SVM for the proposed hybrid classification model after clustering, the f-measure for the Normal class achieves 94.9 %, Denial of Service (DOS) achieves 99.5 %, User to Root (U2R) achieves 55.4 % while Remote to Local (R2L) achieves 98.4 % accuracy which are better than classification metrics using ELM. Using ELM for the proposed hybrid classification model after clustering the Probe class achieves 89.7 % accuracy which is better than SVM. It is thus proposed to use SVM for detecting Denial of Service (DOS), User to Root (U2R), Remote to Local (R2L) and Normal classes while ELM should be used for detecting Probe class.

In summary the main contribution of this thesis can be summarized as follows. Firstly, machine learning algorithms SVM and ELM are utilized for binary and multi class intrusion classification for two benchmark datasets. Secondly, sequential distributed SVM is utilized to reduce time complexity for SVM classifier. Thirdly, a Distributed Neuron Network (DNN) is proposed to reduce server processing time and queuing delay for the ELM classifier. Fourthly, a hybrid scheme consisting of SVM and ELM is utilized to improve the classification for all the intrusion classes. Lastly, an unsupervised machine learning algorithm based on ELM is utilized to classify unlabeled test data and the improvement in classification of various classes using the hybrid classification method is presented.

Acronyms

IoT	Internet-of-Things
SBMS	Smart Building Management System
HAS	Home Automation System
IPS	Indoor Positioning System
AI	Artificial Intelligence
Wi-Fi	Wireless Fidelity
IDPS	Intrusion detection and prevention systems
GUI	Graphical User Interface
RSSI	Received Signal Strength Indicator
MAC	Management and Control Address
NBA	Network Behavior Analysis
HIDS	Host Based Intrusion Detection System
HIS	Hybrid Intrusion Detection System
SD	Signature-based Detection
AD	Anomaly-based Detection
SPA	Stateful Protocol Analysis
SVM	Support Vector Machine

k-NN	k-nearest neighbor
ANN	Artificial Neural Networks
ELM	Extreme Learning Machine
BVM	Ball vector machine
RVM	Relevance Vector Machine
AMI	Advanced Metering Infrastructure
DBN	Deep Belief Networks
SOM	Self Organizing Map
CART	Classification and Regressing Tree
NB	Nave Bayesian Networks
DAG	Directed Acyclic Graph
GA	Genetic Algorithm
EA	Evolutionary Algorithms
MLP	Multilayer perceptron

List of Figures

1.1	Growth Rate Of Cyber Incidents Reported To Computer Emergency Response [1]	1
1.2	Attack sophistication vs. Intruder technical knowledge [1] . .	1
2.1	Types of Intrusion Detection Systems [2]	4
2.2	Taxonomy of Network anomaly detection techniques. [3] . .	6
2.3	Using classification for anomaly detection [4].	7
2.4	Year wise distribution of research papers for the types of classifier design. [5]	8
2.5	SVM for Binary Classification	9
2.6	Irregular classification with SVM	9
2.7	Single Layer Feed Forward Neural Network [6]	10
3.1	An overview of a smart home's architecture, internal and external environments [7]	15
3.2	Home Area Network	16
3.3	Distributed Database Storage on Smart Gateways	16
4.1	Machine Learning Based Intrusion Detection	20
4.2	A Real-Time Network Intrusion Detection System [8]	21
4.3	NSL-KDD Dataset Distribution	25
4.4	ISCX 2012 Dataset Distribution	27
4.5	Multi-Category Classification in LIBSVM [9]	28
4.6	A Single Layer Feed Forward Neural Network (SLFN) with Incremental L2-Norm Direct Solver	42
4.7	Incremental LMS Performance	43

4.8	Network Intrusion Detection System (NIDS) Classification Accuracy	47
4.9	Comparison Between SVM and ELM using 500 Hidden Neurons For Multi Class Classification	47
4.10	Comparison Between SVM and Weighted ELM using 500 Hidden Neurons For NSL-KDD Benchmark Multi Class Classification	48
4.11	Comparison Between SVM and Weighted ELM using 500 Hidden Neurons For ISCX 2012 Benchmark Multi Class Classification	49
5.1	Sequential Distributed SVM (SQ-DSVM)	51
5.2	Distributed Neuron Network with Soft Voting	55
5.3	Flow Diagram For Detecting Intrusions Using DNN	57
5.4	Relationship between Queuing Delay, Total Delay and System Utilization	60
5.5	Multi Level Intrusion Detection System Models [10]	62
5.6	Classification Performance based On Multi SVM and Multi ELM	63
5.7	Hybrid Classification Based On SVM and ELM	64
5.8	Unsupervised ELM Based Hybrid Classification	68

List of Tables

2.1	Total numbers of articles for the types of classifier design using ELM [11]	11
3.1	Smart Home Security Issues [7]	18
4.1	Confusion Matrix	24
4.2	Attack Categories in NSL-KDD Dataset	25
4.3	NSL-KDD Data Preprocessing	26
4.4	ISCX 2012 Experimental set-up parameters	28
4.5	Voting process for 3-category classification [9]	30
4.6	SVM NSL-KDD Binary Classification Confusion Matrices . .	31
4.7	SVM NSL-KDD Binary Classification Performance Metrics .	32
4.8	SVM ISCX 2012 Binary Classification Confusion Matrices .	33
4.9	SVM ISCX 2012 Binary Classification Performance Metrics .	34
4.10	SVM NSL-KDD Multi Class Classification Confusion Matrices	35
4.11	SVM NSL-KDD Multi Class Performance Metrics	36
4.12	SVM ISCX Multi Class Confusion Matrices	37
4.13	SVM ISCX 2012 Multi Class Classification Performance Metrics	38
4.14	ELM NSL-KDD Binary Classification Performance Metrics .	44
4.15	ELM ISCX 2012 Binary Classification Performance Metrics .	45
4.16	ELM NSL-KDD Multi Class Classification Performance Metrics	45
4.17	ELM ISCX 2012 Multi Class Classification Performance Metrics	46
5.1	Training time comparison among different SVM Baselines .	53
5.2	Weighted Average Binary DSN Classification Performance Metrics using 500 Hidden Neurons	58

5.3	Queuing Delay Analysis Between SVM and DNN using 500 Hidden Neurons	61
5.4	F-Measure Comparison between standalone, multilevel and hybrid models (%)	64
5.5	Effect of Increasing Embedding Dimension on Unsupervised ELM Clustering Accuracy	67
5.6	Comparison of Unsupervised ELM against Other Clustering Algorithms [12]	67

Contents

Acknowledgement	i
Abstract	iii
Acronyms	ii
List of Figures	iv
List of Tables	vi
1 Introduction	1
1.1 Intrusion Detection	1
1.2 Thesis Organization	3
2 Literature Review	4
2.1 Types of Intrusion Detection Systems	4
2.2 Intrusion Detection Techniques	5
2.3 Literature Review of SVM in Intrusion Detection	8
2.4 Literature Review of ELM in Intrusion Detection	10
2.5 Literature Review of Intrusion Detection Datasets	13
2.6 Main Categories of Attacks	13
3 IoT Based NIDS Architecture For Smart Home	15
3.1 Smart Home Network Architecture	15
3.2 Security Issues in Smart Home	17
3.3 Security Issues For Distributed Database Storage	19
4 Intrusion Detection Using Machine Learning	20
4.1 NSL-KDD Dataset Description and Pre-Processing	24
4.2 ISCX 2012 Dataset Description and Pre-Processing	27
4.3 Intrusion Detection using SVM	28

4.4	Intrusion Detection using ELM	39
5	Distributed Machine Learning	50
5.1	Intrusion Detection Using Distributed SVM	51
5.2	Intrusion Detection Using Distributed Neuron Network . . .	54
5.3	Multi Level Hybrid Intrusion Detection System Models . . .	62
5.4	Unsupervised Extreme Learning Machine	65
6	Conclusions and Future Work	69
6.1	Conclusions	69
6.2	Future Work	70
	Bibliography	71
A	Publications List	82

Chapter 1

Introduction

1.1 Intrusion Detection

Internet has seen a tremendous amount of growth in its usage over the past few decades. These days many transactions are happening over the Internet since it brings flexibility and ease of use to the consumers. There is a lot of important and valuable personal data which is present online. Organizations and companies holding this data go to great lengths in protecting this information. One of the most common breaches of information which happen are through computer hacking. The rate of computer attacks has risen sharply as shown in Figure 1.1. Due to easier access to hacking tools, it is easier to

Figure 1.1: Growth Rate Of Cyber Incidents Reported To Computer Emergency Response [1]

Figure 1.2: Attack sophistication vs. Intruder technical knowledge [1]

launch computer attacks as shown in Figure 1.2. In the context of smart buildings which employ networked smart gateways, intrusion can severely hamper the performance of different Internet-of-Things (IoT) devices.

Hence anybody using a computer is at some risk of intrusion, even if the computer is not connected to the Internet or any other network (i.e. through physical access). If the computer is left unattended, any person can attempt to access and misuse the system. The problem is, however, far greater if the computer is connected to a network, particularly the Internet. Any user from around the world can reach the computer remotely (to some capacity) and may attempt to access private/confidential information or to launch some form of attack to bring the system to a halt or cease to function effectively [13]. Hence the protection of data and computing resources becomes of paramount importance. Intrusion detection is formally defined as a process in which events occurring in a computer system or a computer network are analyzed for possible intrusions and in which the confidentiality, availability, integrity or other security mechanisms of a computer system are compromised [14]. The aim of intrusion detection is to identify in real time unauthorized misuse of computer systems by attackers both inside and outside the computer system [15]. The researches in this area have developed intrusion detection systems (IDSs) to combat intruders. These systems build a profile of the normal or expected user behavior and then compare this profile to patterns to decide if the profile is normal or abnormal [16]. Intrusion detection techniques fall into three broad categories : Signature-based Detection (SD), Stateful Protocol Analysis (SPA) and Anomaly-based Detection (AD) [17]. In commercial use the SD is the most popular approach because they have very low false alarm rate. But they fall short on detecting new or unknown type of attacks. To combat this issue researches have moved their attention to anomaly based detection which can detect old and new type of attacks. The challenge in using this approach is to minimize false positive rate. Thus the goal in anomaly based detection is to find a classifier which gives high sensitivity, high precision and a very low false positive rate. In this thesis, our focus shall be to use efficient anomaly based machine learning algorithms for intrusion detection based on SVM and ELM.

1.2 Thesis Organization

The thesis is organized as follows: Chapter 2 gives a detailed literature review about this topic and current research which is being done in this area. Chapter 3 gives an overview of Smart Gateway Based Smart Home System. Chapter 4 discusses intrusion detection based on SVM and ELM. Chapter 5 further presents Distributed Support Vector Machine (DSVM) and Distributed Neuron Network (DNN) and elaborates the performance improvement with distributed machine learning. It also discusses hybrid machine learning framework based on SVM and ELM and explores unsupervised extreme learning machine. Section 6 gives conclusion and future work.

Chapter 2

Literature Review

2.1 Types of Intrusion Detection Systems

Various kinds of Intrusion detection and prevention systems (IDPS) exist which are categorized by the type of intrusions they try to recognize and the methods used to identify these intrusions. IDPS fall into three major categories as shown in Figure 2.1:

Figure 2.1: Types of Intrusion Detection Systems [2]

1. *Host Based Intrusion Detection System (HIDS)* : HIDS refers to intrusion detection system (IDS) that monitors activity on a single host system. HIDS collects various information from the host system such as network traffic coming to the host system, log analysis, file changes, policy enforcement, root kit detection, and alerting. HIDS also employ

applications such as firewalls, anti-virus softwares and other spyware detection programs for intrusion detection.

2. *Network Intrusion Detection System (NIDS)* : NIDS on the other hand are placed at appropriate points within the network where there is a linkage between the host computers and external networks. Their objective is to monitor network packets and protect the system from intrusions. To accomplish this, NIDS employs number of sensors to monitor network traffic.
3. *Hybrid Intrusion Detection System (HIS)* : Both HIDS and NIDS have their advantages and disadvantages. By combining multiple classifiers, the hybrid approach aims to reap the benefits of these schemes while overcoming most of their disadvantages.

2.2 Intrusion Detection Techniques

As introduced earlier, research in intrusion detection uses either Signature, Anomaly or Stateful Protocol Analysis. These methods are described below:

1. *Signature Based Detection* : Every threat will have a unique signature. In this scheme, the footprint or signature of an intrusion is matched against a known set of threats in a database. This method is highly effective in discovering known attacks. However this method is highly ineffective in discovering unknown attacks. With the ever growing number and evolution of intrusions it is thus necessary to detect unknown attacks as well.
2. *Anomaly Based Detection* : Anomaly based detection is based on the principal that abnormal behavior would be significantly different and observable from the normal behavior of the user. Normal profiles are built over a period of time and a set of rules are set. Any network traffic which does not conform to these rules is considered as an intrusion. This method can be very effective in detecting unknown attacks. However this technique suffers from high false positives. Thus any system which employs this technique should have a low false positive rate.

3. *Stateful Protocol Analysis* : In this method predetermined profiles of intrusive traffic for each protocol is compared against observed traffic to identify abnormalities [18].

Figure 2.2: Taxonomy of Network anomaly detection techniques. [3]

In this work, the focus is on anomaly based detection of intrusions. Anomaly based detection can be done using Classification, Statistical, Clustering or by using information theory [3] as shown in Figure 2.2. In particular we use Classification based detection of network intrusions.

2.2.1 Anomaly Detection Using Classification Techniques

In this approach a classifier is trained from a labeled training data set to form a model in the training phase. During the testing phase, the classifier in conjunction with the learned model tries to classify the test instance into a normal or as an intrusion [19] [20]. This process is shown in Figure 2.3. The labeled data can be multi - dimensional i-e have more than one label belonging to normal classes in which case the classification will be multi-class. In such a detection scheme the classifier tries to distinguish between each normal class and the rest of the classes as shown in Figure 2.3 (a). In case where the labeled data have only a single label the classification will be one class anomaly detection as shown in Figure 2.3 (b). In such a

classification, the classifier tries to learn a discriminative boundary around the normal instances using a one-class classification algorithm such as one-class SVMs [4].

Figure 2.3: Using classification for anomaly detection [4].

In this scheme, it is assumed that a classifier can distinguish between normal and abnormal network traffic in the feature space. Research Literature has used single, hybrid or an ensemble of these classifiers which are described below.

Single classifiers

Single Classifiers are defined where only a single machine learning algorithm is used to classify network intrusions. Some of the classifiers used in research literature are k-nearest neighbor (k-NN), SVM, Artificial Neural Networks (ANN), Decision trees, Naive Bayes networks, Genetic Algorithm and Fuzzy logic [11].

Hybrid classifiers

Hybrid classifiers use one or more single classifiers for intrusion detection. They are primarily used to increase overall accuracy of the system. Some of the techniques used in hybrid classifiers are based on cascading different classifiers, clustering based approaches or using the classifiers in the way in which the first classifier tries to optimize the performance of the second classifier [11].

Ensemble classifiers

Ensemble classifiers use combination of several weak learners. These classifiers are trained on a subset of the training samples to achieve a better overall classification accuracy. Some of the methods used to achieve this are majority voting, boosting and bagging [11].

In one of the most recent surveys on using machine learning for intrusion

Figure 2.4: Year wise distribution of research papers for the types of classifier design. [5]

detection, Haq *et.al* [5] have shown the distribution of papers using single, ensemble and hybrid machine learning techniques. Figure 2.4 shows this distribution from 2009 to 2014.

2.3 Literature Review of SVM in Intrusion Detection

In this section, a detailed literature review regarding use of SVM in intrusion detection is given. SVM was proposed by Vapnik [21]. SVM is an example of a supervised machine learning algorithm and builds predictors by analyzing training data, looking at patterns which are used for classification or regression. In the training phase, it tries to build a model which is representative of the gap as shown in Figure 2.5 which will divide categories of dataset. By maximizing the margin plane, the two categories can be separated as much as possible. The model will then be used for testing data to predict which category each instance belongs to. By comparing the actual and predicted category, classification accuracy can be found.

In addition to performing binary classification, SVM can also be used to

Figure 2.5: SVM for Binary Classification

Figure 2.6: Irregular classification with SVM

perform multi class classification as shown in Figure 2.6(a). It achieves this using a non-linear classification technique, the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces as shown in Figure 2.6(b). The kernel functions can be Linear, polynomial, RBF or Sigmoid. In their survey Haq *et.al* [5] reported that the most commonly used intrusion detection technique was SVM. Mukkamala *et.al* [22] used SVM and neural networks for intrusion detection and used KDDCUP99 DARPA dataset [23]. In [24], the authors observed that the KDDCUP99 dataset had redundant records. In [25], SVM and neural networks were used for extracting important features and showed a high detection accuracy. In [26], effect of various SVM kernel functions on the accuracy of intrusion classification was performed. In [27], SVM was used to monitor Voice Over IP (VoIP) to distinguish normal and abnormal connections. In [28], training time complexity for using SVM was shown to be improved using hierarchical clustering analysis. In [29], SVM and ANN was used to classify host audit data to detect abnormalities. In [30], Robust SVM (RSVM) was used to counter noise in the training data. In [31], an intrusion detection system (IDS) based on

SVM and hierarchical clustering was proposed. In [32], an autonomous labeling approach was presented. In [33], one class classification method using SVM was proposed.

2.4 Literature Review of ELM in Intrusion Detection

ELM was developed as a new classifier which randomly selects the hidden nodes and determines the output weights of a Single Layer FeedForward Neural Network (SLFN) [34] analytically. Figure 2.7 shows a SLFN. The first layer is the input layer, the second layer is the hidden layer and last layer is the output layer. No parameter tuning of the hidden layer is necessary and the input weights and biases are chosen randomly. Between the input layer and the hidden layer, input weights are formed, the thresholds for the hidden layer are the biases whereas between the hidden layer and the output layer output weights are formed. ELM has been widely used as a single, hybrid or in ensemble of classifiers since its inception. Table 2.1 gives an overview of the research papers which have used ELM.

Figure 2.7: Single Layer Feed Forward Neural Network [6]

1. *Single Classifier:* Huang *et.al* proposed ELM for intrusion detection [35]. They showed the comparative advantage of basic ELM in training and testing time compared to SVM and achieved greater accuracy using kernalized ELM in comparison to SVM. Giles [36] and Singh [39] [40] used Online Sequential ELM (OS-ELM) to reduce time

Table 2.1: Total numbers of articles for the types of classifier design using ELM [11]

Classifier design Type	No. of Research paper	References
Single	11	(Huang, 2012 [35]), (Gilles, 2012 [36]), (Jaiganesh, 2012 [37]), (Gideon, 2012 [38]), (Xiang, 2014 [6]), (Singh, 2015 [39]), (Singh, 2015, [40]), (Intarasothonchun, 2015, [41]), (Srimuang, 2015, [42]), (Alom, 2015, [43]), (Li, 2016 [44])
Hybrid	9	(Cai, 2012, [45]), (JaiGanesh, 2014, [46]), (Zhang, 2014, [47]), (Akusok, 2014, [48]), (Pan, 2014, [49]), (Jayaprakash, 2015, [50]), (Raj, 2015, [51]), (Jayabalan, 2016, [52]), (Ma, 2016, [53])
Ensemble	5	(Jingjing, 2015, [54]), (Bosman, 2015, [55]), (Zhifan, 2015, [56]), (Shukla, 2015, [57]), (Yu, 2016, [58])

complexity. Jaiganesh [37] showed benefits in improved accuracy and detection rate by using kernelized ELM compared to basic ELM. Similarly Gideon [38] discussed benefits of reduced training time by using ELM. Xiang [6] used ELM in a big data environment and used map reduce framework for their research. Intarasothonchun [41] and Srimuang [42] used weighted ELM for intrusion detection. Alom [43] used Deep Belief Networks (DBN) for intrusion detection and compared performance against ELM. Li [44] researched the use of ELM for Advanced Metering Infrastructure (AMI) in Smart Grid.

2. *Hybrid Classifier*: Cai [45] used a fusion of Ball vector machine (BVM) and ELM for intrusion detection and claimed that by using this fusion method the classification accuracy in terms of sensitivity and false alarm rate is similar with a much reduced training time. JaiGanesh [46] proposed a new intrusion detection system that is formed by using modified classification process of SVM, ELM and Relevance Vector Machine (RVM). They showed that this scheme can achieve higher

sensitivity and low false rate than the regular kernelized SVM and Fast Hierarchical ELM algorithms. Zhang [47] proposed to use artificial bee colony (ABC) to optimize the kernelized ELM and showed that the proposed ABC-KELM can detect the network intrusion efficiently and its performance is superior to unoptimized kernelized ELM method. Ma [53] proposed a fusion of gaussian kernels as the kernel function of ELM where the kernel related parameters and the weights are optimized using self-adaptive artificial bee colony (SABC) and claimed superior performance than several related methods. Akusok [48] used a two stage decision methodology using k-NN and ELM and showed that this method is able to identify new samples quickly. Pan [49] proposed integration of multi-kernel function based kernel direct discriminant analysis (MKDDA) and quantum particle swarm optimization (QPSO) optimized kernel extreme learning machine. They claimed that using the proposed technique achieved better detection accuracy and training speed. Jayaprakash [50] used a hybrid of Levenberg-Marquardt optimization and kernelized ELM for intrusion detection and achieved satisfactory performance for detection accuracy and training time. Raj [51] proposed to use N-gram extraction algorithm for a new host based intrusion detection system (IDS) using Improved Extreme Learning Machine (I-ELM), to simultaneously reduce the training complexity and increase the sensitivity. Jayabalan [52] proposed a hybrid extreme learning machine classifier to detect the network intrusions in a network.

3. *Ensemble Classifier*: Bosman [55] combined many classifiers in an ensemble, to increase the anomaly detection accuracy and thus boosting the performance of the individual classifiers. Shukla [57] proposed a voting based ELM in which a majority voting based ensemble technique was used and claimed better performance. Yu [58] proposed an efficient cascaded classifier using ELM for intrusion detection which are serially connected.

2.5 Literature Review of Intrusion Detection Datasets

In their survey, Shelly *et.al* [59] mentioned that data came from three sources which includes parsing network data packets, sequence of commands from the user input and system state information such as memory usage, log files or call sequences. Ahmed *et.al* [3] mentioned few data sets are publicly available due to privacy concerns. Darpa/KDD datasets are considered as benchmark in the research literature. McHugh [60] criticized the use of Darpa benchmark. Some of the dataset repositories reported by Ahmed *et.al* [3] which are used in research include : *PREDICT* [61], *CAIDA* [62], *DEFCON* [63], *ADFA Intrusion Detection Datasets* [64], *NSL-KDD* [65], *KYOTO* [66], *ISCX 2012* [67], *ICS Attack* [68] For this research NSL-KDD [65] and ISCX 2012 [67] dataset were chosen as a benchmark for network intrusion.

2.6 Main Categories of Attacks

There are 4 major types of categories of attacks reported in the 1998 DARPA intrusion detection evaluation dataset [69].

1. *Denial of Service (DOS)* : In the denial of service [69] attack the memory and computing resources are flooded with illegitimate requests in such a way that the actual users are denied service to the machine. A distributed denial of service (DDOS) [70] attack is one which is carried out by bots using automated tasks.
2. *Probe* : In the probing attack an attacker tries to gather information about the system or the network. This type of attack is carried out by distributing number of programs to gather information or know some vulnerabilities about the system [69].
3. *Remote to Local (R2L)* : This type of attack happens when a remote user who is not entitled to system privileges tries to get access and is a privilege escalation attack. It is called a Remote to Local attack

because the attacker tries to get unauthorized access from a remote machine i-e by guessing password. The attacker tries to break into the system by exploiting the weakness of the system. This is specially true when users use common passwords which are guessed easily [70].

4. *User to Root (U2R)* : This type of attack happens when the attacker with normal user privileges wants to resume root privileges of a system with the most frequent attack being the buffer overflow attack [69]. It is called a U2R attack because the attacker tries to gain local superuser root privileges of a machine.

Chapter 3

IoT Based NIDS Architecture For Smart Home

3.1 Smart Home Network Architecture

Figure 3.1: An overview of a smart home's architecture, internal and external environments [7]

Figure 3.1 shows the architecture of a smart home and the interaction it has with its internal and external environments. The internal environment of the smart home consists of all the appliances and devices which are centrally controlled by the Energy Management System (EMS). EMS is responsible

Figure 3.2: Home Area Network

Figure 3.3: Distributed Database Storage on Smart Gateways

for management of various appliances and systems inside the smart home so that it can adapt its energy profile to suit the grid's capabilities [7]. These entities are represented by red-dotted lines in figure 3.1. The network inside a smart home makes a home area network (HAN). It is established among home appliances and smart meter. Zigbee (802.15.4) with data rates up to 250 kbps is a common wireless standard used inside a Home Area Network (HAN) [71] to establish a communication network.

As such our Home Area Network (HAN) consists of the following components:

1. *Smart gateways*: These are IoT devices at the heart of smart homes. We utilize Beagleboard-Xm [72] as our smart gateway. Beagleboard-Xm runs on Ubuntu 14.04 LTS with an ARM processor clocked at 1 GHz and 512 MB RAM. The advantage of using these boards is that they are open source hardware and are designed to deliver previous generation laptop like performance to an improved level. There is facility of on board storage which can eliminate the need for a centralized server. There are four external USB ports which can support useful extensions such as Zigbee extension in fast connection for home appliances in home automation system. Another advantage of using these boards is that the ubuntu operating system is open source. Ubuntu has advantages over other embedded operating systems in that the consumer does not need to pay for licensing fee or any royalties. It

is also a very stable kernel and the users have the ability to modify and redistribute the source code since it is an open source project [73]. These smart gateways also use several sensors for capturing information such as light intensity, temperature, humidity and Wi-Fi data.

2. *Smart meters*: Smart meters collect current information from home appliances and are able to be remotely controlled by Smart Devices. We have built a GUI based application on an android platform which can remote control these smart meters as well as provide the user with real time power consumption.
3. *Smart sensors*: These are able to collect environment information such as light intensity, temperature, humidity, Wi-Fi data.
4. *Smart sockets*: Smart sockets collect current information from home appliances and are able to be remotely controlled by Smart Devices.
5. *Smart devices* : These are the devices used by the user directly and contain a Graphical User Interface (GUI) based on android platform to remotely control smart meters.
6. *HAN Intrusion Detection System (IDS) module*: is the IDS module at the HAN level which tracks the network traffic for intrusions [71].

Communication layers external to the HAN include Neighborhood Area Network (NAN) and the Wide Area Network (WAN). Both NAN and WAN include IDS modules for detecting intrusions.

3.2 Security Issues in Smart Home

Table 3.1 shows the different types of threats which can affect the smart home. The threats can occur at the networking domain (N) or derived by a smart home (SH) introduced concept [7]. Specifically each scenario is described as follows:

1. *SH₁: Attacks Threatening Successful Device Energy-Consumption Reporting*: In this type of attack during the transmission of data from

Table 3.1: Smart Home Security Issues [7]

Scenario Num:	Possible Threads	Security Goals Compromised	Degree of Impact
SH ₁	Eavesdropping (N) Traffic Analysis (N) Message Modification (N) Replay Attack (N) EMS Impersonation (SH)	Confidentiality Integrity Authenticity	L-M
SH ₂	Repudiation (N) Message Modification (N) Replay Attack (N)	Non repudiation Integrity Authenticity	M
SH ₃	Tampering/Reversal Removal of Meter (SH) Illegal Software Modification/Update (SH)	Authenticity Integrity	M
SH ₄	Customer Impersonation (N) Device Impersonation (SH) Message Modification (N) Replay attack (N) Repudiation (N)	Integrity Non repudiation Authentication	L-H
SH ₅	Customer Impersonation (N) Eavesdropping/Message (N) Interception (N) Message Modification (N)	Integrity Non repudiation Authentication	L-M

appliances to the EMS, an eavesdropping attack can occur which can result in valuable data leakage to the adversary [7].

2. *SH₂: Attacks Aiming Energy Import/Export Signals at the ESI/HAN:* In this type of attack an attacker can impersonate ESI/HAN and receive signals from the grid so that they never reach the smart home. Similarly the attacker can modify the contents of a message [7].
3. *SH₃: Physical meter tampering/reversal or removal:* Customers with malevolent intentions try to remove/reverse the meter or alter its software to have a reduced electricity bill [7].
4. *SH₄: Attacks Against Remote Home Monitoring and Control:* In this type of attack the attacker can impersonate the client to remotely control the appliances inside a smart home. The attacker can also replay certain messages so that some appliances operate on regular intervals [7].
5. *SH₅: Attacks Aiming the Requests for Energy Usage Data:* In a smart home customers can request to receive their energy consumption profile. Attackers can impersonate the client to gain an insight into the

consumption of energy or eavesdrop on request signals [7].

3.3 Security Issues For Distributed Database Storage

We have developed a distributed database storage platform using Cassandra on smart gateways. This nosql based database storage is vulnerable to several intrusions as well. Cassandra data files are kept unencrypted and no mechanism exists to encrypt data automatically once it has been stored. This can result in the attacker to access the file system and extract confidential information. To avoid this the application should encrypt the data before storing in the database [74]. By default inter cluster communication is not encrypted and authenticated and nodes can talk to each other freely. To counter this all the cluster members should have their encryption enabled and all the client certificates should be installed. [74]. Cassandra query language is also vulnerable to SQL like injection attacks and applications should use prepared statements for storing data [74]. Cassandra can also suffer from a Denial of service (DOS) attack and to prevent this connection pooling should be used [74].

Chapter 4

Intrusion Detection Using Machine Learning

In machine learning approach for network intrusion detection system (NIDS), the detection for intrusion can be interpreted as a binary classification problem, to classify normal and attack instances. In the same way intrusion can also be considered as a multi-class classification problem where intrusion instances belong to a specific class of an attack. We can use supervised, semi supervised or unsupervised machine learning approach to achieve our objective. Figure 4.1 shows the network intrusion detection using supervised machine learning algorithms such as SVM and ELM.

Figure 4.1: Machine Learning Based Intrusion Detection

Intrusion Detection is defined as a process in which events occurring in a computer system are analyzed which might be violating the computer security policies [18]. To detect and prevent these intrusions several intrusion

Figure 4.2: A Real-Time Network Intrusion Detection System [8]

detection systems IDSs are deployed at the Home Area Network (HAN), Neighborhood Area Network (NAN) and Wide Area Network (WAN) in the smart grid as previously mentioned. These modules are capable of processing incoming traffic and intrusions can be classified by anomaly based machine learning algorithms such as SVM and Single Hidden Layer Feed Forward Neuron Network (SLFN). Figure 4.2 shows a real time network intrusion detection system [8]. In the pre-processing phase raw packets are captured by a packet sniffer in order to extract packet information such as TCP header, ICMP header, UDP header and IP header from each packet. The traffic is then captured for a certain amount of time and then data features are used as signatures for that network traffic [8]. Using this approach Denial of service (DOS) and Probe attacks can be found [8]. Data for attacks relating to Root to Local (R2L) can be collected by analyzing log files from the host computer to check for number of failed login attempts and to check for

network parameters such as duration of connection and the type of service requested where as User to Root (U2R) attacks involve checking for number of files which were created and the shell prompts invoked [75].

As such the main objectives are :

1. **Objective 1:** Achieve an overall high accuracy and detection rate. Following terms are defined to mathematically formulate the objectives.

- (a) *False Positives (FP)*: Normal instances which are detected as intrusions.
- (b) *False Negatives (FN)*: Intrusion instances which are detected as normal.
- (c) *True Positives (TP)*: Correctly detected intrusion instances.
- (d) *True Negatives (TN)*: Correctly detected normal instances.

Several statistical measures are used to categorize the performance for Binary Network Intrusion Detection System (NIDS). Specifically following measures are used to characterize the performance.

- (a) *Recall/Sensitivity*: is a measure of detection rate or recall of the system to detect attacks and is defined as :

$$Sensitivity = \frac{TP}{TP + FN} \times 100 \quad (4.1)$$

- (b) *False Positive Rate (FP)*: gives a measure of fall out or false positive rate i-e., normal instances being classified as intrusions.

$$FP = \frac{FP}{FP + TN} \times 100 \quad (4.2)$$

- (c) *Precision* : Precision is a measure of predicted positives which are actual positives.

$$Precision = \frac{TP}{TP + FP} \times 100 \quad (4.3)$$

- (d) *F-Measure* : F-Measure is a metric that gives the harmonic mean of precision and sensitivity.

$$F - Measure = \frac{2}{\frac{1}{precision} + \frac{1}{Sensitivity}} \times 100 \quad (4.4)$$

- (e) *Matthews Correlation Coefficient (MCC)* : MCC measures the quality of binary classification. It represents values in the range -1 to +1. A value of +1 represents 100 % prediction, -1 represents 0 % prediction. A value of 0 represents a classification which is no better than random prediction.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (4.5)$$

- (f) *Overall Accuracy*: Overall accuracy is defined as a ratio of TP and TN to the total number of instances

$$Accuracy = \frac{TP + TN}{FP + FN + TP + TN} \times 100 \quad (4.6)$$

- (g) *Confusion Matrix*: The confusion matrix contains information about the classification done. It displays information regarding distinct classes which have to be classified. It displays information in a matrix form about the actual class labels against the predicted class labels. It contains numbers for true positives, true negatives, false positives and false negatives. These relationship are shown below in table 4.1

Table 4.1: Confusion Matrix

		Predicted Value	
		positive	negative
Actual Value	positive	True Positive	False Negative
	negative	False Positive	True Negative

2. **Objective 2:** Reduce the training and testing time on resource limited Hardware BeagleBoard-xM. To distribute training task on gateways with n number, the average training time should be minimized to reflect the reduced complexity on such gateway system. This is important since Beagleboard-xM makes up the analyzing module inside the Home Area Network (HAN).

$$\min \frac{1}{n} \sum_{i=1}^n t_{train,i} \quad (4.7)$$

$s.t. \ e < \epsilon$

3. **Objective 3:** Reduce server load and detection latency for intrusion detection modules at various network layers in the smart grid so that alarms can be raised quickly to protect the system.

4.1 NSL-KDD Dataset Description and Pre-Processing

For this research NSL-KDD dataset [65] was used. For Binary classification KDDTrain+20Percent.arff dataset was used in training phase and for testing phase KDDTest+.arff was used.

Figure 4.3: NSL-KDD Dataset Distribution

KDDTrain+20Percent.arff contains 25192 instances where as KDDTest+.arff contains 22544 instances. For multi class classification, KDDTrain+20Percent.txt and KDDTest+.txt was used which contains the attack groups as mentioned in table 4.2. This distribution is shown in Figure 4.3.

Table 4.2: Attack Categories in NSL-KDD Dataset

Category	Type
DOS	back, land, neptune, pod, smurf, teardrop mailbomb, processtable, apache2, udpstorm
PROBE	ipsweep, nmap, satan, portsweep, mscan, saint
U2R	perl, buffer_overflow, loadmodule, rootkit, sqlattack, xterm, ps
R2L	ftp_write, guess_passwd, snmpgetattack, imap, multihop httptunnel, phf, spy, warezclient, warezmaster, worm snmpguess, named, xsnoop, xlock, sendmail

There are a total of 41 features in the dataset. Of these 38 are numerical features and 3 are symbolic features [59]: In order to use either SVM or ELM on Beagleboard-Xm we need to convert symbolic features to numeric features. So some form of coding is necessary to convert the symbolic features to numeric features. The protocol type takes the symbolic values of tcp, udp or icmp. Similarly the service field take 70 different symbolic values where as the flag field takes 11 different symbolic values. Hernandez *et.al* [76] described various techniques which can be used to convert symbolic features to numeric features. These techniques include :

1. *Arbitrary assignment*: In this scheme, a sequence of integers is used

Table 4.3: NSL-KDD Data Preprocessing

Feature	Numeric-valued Transformation
protocol type	tcp = 1, udp = 2, icmp = 3
service	aol=1, auth=2, bgp=3, courier=4 csnet_ns=5, ctf=6, $\dots$, Z39_50 = 70
flag	OTH=1, REJ=2, RSTO=3, RSTOS0=4 RSTR=5, S0=6, S1=7, S2=8, S3=9, SF=10, SH=11
IDS(2)	normal=1, attack=2,
IDS(5)	normal=1, probe=2, dos=3, u2r=4, r2l=5

to represent symbolic features.

2. *Indicator variables:* This coding uses binary coding to represent presence or absence of a feature. This can increase the dimensionality of the data and cause issues with classifiers [76].
3. *Conditional probabilities:* In this scheme each symbolic feature is replaced with an array of conditional probabilities to obtain a class given the attribute a given class has.
4. *SSV Criterion based method:* The Separability Split Value (SSV) criterion is used for decision tree construction. This method uses a split value which produces a subset of the set of alternative values of one feature [76]

In this research we use sequential arbitrary assignment coding scheme to convert symbolic features to numeric features. Table 4.3 shows this assignment. In Table 4.3, ID(2) represents binary classification where normal class was coded with 1 and attack instance was coded 2. On the other hand, ID(5) represents multi classification representing normal and the 4 attack groups in Table 4.2.

Figure 4.4: ISCX 2012 Dataset Distribution

4.2 ISCX 2012 Dataset Description and Pre-Processing

ISCX 2012 Dataset [77] was developed at the University of Brunswick ISCX. The original dataset contains 19 features and a label representing Normal instances and intrusions belonging to Denial Of Service (DOS), brute force secure shell (SSH) attacks, internal infiltrations (L2L) and Botnet (Distributed Denial Of Service DDOS) intrusions. Some of the features were irrelevant and were removed from the dataset. Additionally some of the features in the data set i.e., appName, direction, sourceTCPFlagsDescription, destinationTCPFlagsDescription and protocolName were symbolic in nature and an arbitrary sequential integer assignment was used to convert these features to numeric features similar to NSL-KDD benchmark. The 11 features which are relevant and present in the dataset were appName, totalSourceBytes, totalDestinationBytes, totalDestinationPackets, totalSourcePackets, direction, sourceTCPFlagDescription, destinationTCPFlagDescription, protocolName, sourcePort and destinationPort. Due to the large size of the dataset a small subset was chosen to represent the training and the testing datasets. Figure 4.4 shows the ISCX 2012 distribution of data for binary and multi class classification. Table 4.4 gives the description of ISCX 2012 benchmark for intrusion detection.

Table 4.4: ISCX 2012 Experimental set-up parameters

Parameter	Value
Training Data Size	103285
Testing Data Size	103229
Data Dimension	11
Number of Labels	Binary (2)
	MultiClass (5)
Number of Gateways	5

4.3 Intrusion Detection using SVM

Figure 4.5: Multi-Category Classification in LIBSVM [9]

In this section performance analysis is done on the NSL-KDD benchmark using SVM for both Binary and Multi class classification. SVM has been extensively used for different classification tasks because of its reliability and high accuracy. The main idea behind SVM in intrusion detection is to separate two points classes of a training data set [78].

$$D = \{(x_i), (y_i), i = 1, \dots, N, x_i \in R^n, y_i \in \{-1, +1\}\} \quad (4.8)$$

The problem formulation is to obtain a decision function $F : R^n \rightarrow \{-1, +1\}$

which can be achieved by solving the convex quadratic optimization problem:

$$\min f(\alpha) = \frac{1}{2}\alpha^T Q \alpha - e^T \alpha \quad (4.9)$$

such that $0 \leq \alpha_i \leq C, i = 1, \dots, l$ and $y^T \alpha = 0$. Here $y = [y_1, y_2, \dots, y_N]^T$, $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_N]^T$ and C is a constant. e is a vector of all 1's. Q is a symmetric positive semidefinite matrix whose $Q_{i,j}$ entries are defined as :

$$Q_{i,j} = y_i y_j K(x_i, x_j), i, j = 1, 2, \dots, N \quad (4.10)$$

Here $K(x_i, x_j)$ represents a kernel function. There are four common kernel functions:

1. *Linear*

$$K(x_i, x_j) = x_i \cdot x_j \quad (4.11)$$

2. *Polynomial Kernel function:*

$$K(x_i, x_j) = [(x_i \cdot x_j) + 1]^p \quad (4.12)$$

where p is the dimension and $p \geq 1$.

3. *Radial Basis Function (RBF) Kernel function:*

$$K(x_i, x_j) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right) \quad (4.13)$$

where σ is the kernel width.

4. *Sigmoid kernel function:*

$$K(x_i, x_j) = \tanh[v(x_i \cdot x_j) + c] \quad (4.14)$$

For determining which class an instance belongs to, the decision function should be $F : R^n \rightarrow \{-1, +1\}$ in case of binary classification. To get the decision function in the classification problem, LIBSVM [79] based on Sequential Minimum Optimization-type SVM (SMO-SVM) is a popular SVM

solver and is chosen for multi category intrusion detection classification problem. In case of intrusion detection different classes represent different types of intrusions. In LIBSVM multi-category classification uses one versus one (1-v-1) SVM. If there are k classes, the solution follows to solve $k(k-1)/2$ binary problems. In the prediction phase for a 3 class problem it will gener-

Table 4.5: Voting process for 3-category classification [9]

Category	Binary Classifier			Votes	Result
	1 vs 2	1 vs 3	2 vs 3		
Class 1	Win	Win	X	2	Win
Class 2	X	X	Win	1	X
Class 3	X	X	X	0	X

ate 3 binary classifiers as shown in Figure 4.5. For a new input instance, it will pass through all the binary classifiers and a voting process will decide which class the intrusion instance belongs to. The process is shown in Table 4.5. In the first case the instance will pass through the 1 vs 2 binary classifier and if it belongs to Class 1 it will get 1 vote. In the next phase it will move on to 1 vs 3 binary classifier and again if it belongs to Class 1 it will get 1 vote. So here class 1 gets 2 votes. Same process is repeated for the other 2 classes and at the end if class 1 gets the most votes so the dataset will be classified as belonging to class 1.

4.3.1 Binary Classification

In this section binary classification results are presented using the four kernel functions for NSL-KDD and ISCX intrusion detection benchmarks.

NSL-KDD Benchmark Results

Tables 4.6a, 4.6b, 4.6c and 4.6d show the confusion matrices for the NSL-KDD data. Based on these confusion matrices Recall, FP rate, precision, F-

Table 4.6: SVM NSL-KDD Binary Classification Confusion Matrices

(a) SVM-Linear Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	8974	737
	<i>anomaly</i>	4875	7958

(b) SVM-Polynomial Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	9395	316
	<i>anomaly</i>	5021	7812

(c) SVM-Sigmoid Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	8978	733
	<i>anomaly</i>	5169	7664

(d) SVM-RBF Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	8974	737
	<i>anomaly</i>	5255	7578

Measure, MCC and Accuracy performance metrics are calculated for normal and anomaly classes. The weighted average for a performance metric is calculated, as the ratio of the sum of the product of each performance metric to the number of instances in each class to the total number of instances.

The performance metrics are displayed in tables 4.7a - 4.7d. In addition the overall accuracy, training and testing times are shown in table 4.7e. For binary classification, polynomial kernel gives the best classification in terms of overall accuracy. However it requires twice the training and testing times.

Table 4.7: SVM NSL-KDD Binary Classification Performance Metrics

(a) SVM-Linear Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.924	0.380	0.648	0.762	0.554
anomaly	0.620	0.076	0.915	0.739	0.554
Weighted Avg.	0.751	0.207	0.800	0.749	0.554

(b) SVM-Polynomial Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.967	0.391	0.652	0.779	0.594
anomaly	0.609	0.033	0.961	0.745	0.594
Weighted Avg.	0.763	0.187	0.828	0.760	0.594

(c) SVM Sigmoid Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.925	0.403	0.635	0.753	0.534
anomaly	0.597	0.075	0.913	0.722	0.534
Weighted Avg.	0.738	0.216	0.793	0.735	0.534

(d) SVM RBF Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.924	0.409	0.631	0.750	0.528
anomaly	0.591	0.076	0.911	0.717	0.528
Weighted Avg.	0.734	0.220	0.790	0.731	0.528

(e) Performance Metrics Using SVM Kernels For Binary Classification

SVM Kernel Type	Accuracy %	TrainingTime (s)	TestingTime (s)
Linear	75.11	76.52	5.28
Polynomial	76.32	183.25	13.52
RBF	73.42	84.54	6.91
Sigmoid	73.82	102.6	8.06

ISCX 2012 Benchmark Results

Tables 4.8a, 4.8b, 4.8c and 4.8d show the confusion matrices for the ISCX 2012 data.

Table 4.8: SVM ISCX 2012 Binary Classification Confusion Matrices

(a) SVM-Linear Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	69039	751
	<i>anomaly</i>	285	33154

(b) SVM-Polynomial Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	66222	3568
	<i>anomaly</i>	1017	32422

(c) SVM-Sigmoid Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	69005	785
	<i>anomaly</i>	399	33040

(d) SVM-RBF Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	69006	784
	<i>anomaly</i>	511	32928

The performance metrics are displayed in tables 4.9a - 4.9d

The overall accuracy, training and testing times on PC are shown in table 4.9e. For this dataset Linear kernel gives the best overall performance metrics in terms of accuracy, training and testing times. In order to choose a better kernel, multi class classification is performed to test the performance of the classifier on different classes for both NSL-KDD and ISCX 2012 benchmarks.

Table 4.9: SVM ISCX 2012 Binary Classification Performance Metrics

(a) SVM-Linear Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.989	0.009	0.996	0.993	0.977
anomaly	0.991	0.011	0.978	0.985	0.977
Weighted Avg.	0.990	0.009	0.990	0.990	0.977

(b) SVM-Polynomial Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.949	0.030	0.985	0.967	0.902
anomaly	0.970	0.051	0.901	0.934	0.902
Weighted Avg.	0.956	0.037	0.958	0.956	0.902

(c) SVM Sigmoid Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.989	0.015	0.993	0.991	0.971
anomaly	0.985	0.011	0.977	0.981	0.971
Weighted Avg.	0.987	0.014	0.987	0.987	0.971

(d) SVM RBF Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.989	0.012	0.994	0.991	0.974
anomaly	0.988	0.011	0.977	0.982	0.974
Weighted Avg.	0.989	0.012	0.989	0.989	0.974

(e) Performance Metrics Using SVM Kernels For Binary Classification

SVM Kernel Type	Accuracy %	TrainingTime (s)	TestingTime (s)
Linear	98.99	3401.41	20.85
Polynomial	95.55	6600.56	749.79
RBF	98.85	4043.06	67.35
Sigmoid	98.74	4760.96	89.99

4.3.2 Multi Class Classification

In this section multi class classification is performed on the two benchmark datasets and results are presented using the four kernels.

NSL-KDD Benchmark Results

Table 4.10: SVM NSL-KDD Multi Class Classification Confusion Matrices

(a) SVM-Linear Confusion Matrix

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>Probe</i>	<i>U2R</i>	<i>R2L</i>
Actual Class	<i>Normal</i>	8992	90	621	3	5
	<i>DOS</i>	1482	6135	21	0	0
	<i>Probe</i>	634	154	1627	0	6
	<i>U2R</i>	41	0	0	21	5
	<i>R2L</i>	2395	3	5	5	299

(b) SVM-Polynomial Confusion Matrix

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>Probe</i>	<i>U2R</i>	<i>R2L</i>
Actual Class	<i>Normal</i>	9001	86	617	3	4
	<i>DOS</i>	2025	5591	22	0	0
	<i>Probe</i>	547	212	1662	0	0
	<i>U2R</i>	37	0	0	25	5
	<i>R2L</i>	2389	2	1	5	310

(c) SVM-RBF Confusion Matrix

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>Probe</i>	<i>U2R</i>	<i>R2L</i>
Actual Class	<i>Normal</i>	9600	7	104	0	0
	<i>DOS</i>	2414	5138	86	0	0
	<i>Probe</i>	420	362	1639	0	0
	<i>U2R</i>	67	0	0	0	0
	<i>R2L</i>	2540	5	130	0	32

(d) SVM-Sigmoid Confusion Matrix

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>Probe</i>	<i>U2R</i>	<i>R2L</i>
Actual Class	<i>Normal</i>	9010	265	429	3	4
	<i>DOS</i>	1911	5704	23	0	0
	<i>Probe</i>	467	220	1734	0	0
	<i>U2R</i>	36	0	0	26	5
	<i>R2L</i>	2365	4	1	5	332

The resulting performance metrics resulting from these confusion matrices are calculated for Normal, DOS, Probe, U2R and R2L classes and are shown in tables 4.11a - 4.11e. It can be observed that Sigmoid Kernel function gives a better detection rate compared to other kernel functions.

Table 4.11: SVM NSL-KDD Multi Class Performance Metrics

(a) SVM-Linear Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.926	0.355	0.664	0.773	0.578
DOS	0.803	0.017	0.961	0.875	0.826
Probe	0.672	0.032	0.715	0.693	0.658
U2R	0.313	0.00	0.724	0.438	0.475
R2L	0.110	0.001	0.949	0.198	0.304
Weighted Avg.	0.757	0.162	0.805	0.729	0.637

(b) SVM-Polynomial Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.927	0.389	0.643	0.759	0.549
DOS	0.732	0.020	0.949	0.827	0.767
Probe	0.686	0.032	0.722	0.704	0.669
U2R	0.373	0.00	0.758	0.500	0.531
R2L	0.115	0.00	0.972	0.205	0.314
Weighted Avg.	0.736	0.178	0.795	0.709	0.607

(c) SVM-RBF Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.989	0.424	0.638	0.776	0.593
DOS	0.673	0.025	0.932	0.781	0.713
Probe	0.837	0.016	0.837	0.748	0.727
U2R	0.00	0.00	0.00	0.00	0.00
R2L	0.012	0.00	1.00	0.023	0.102
Weighted Avg.	0.728	0.193	0.801	0.682	0.587

(d) SVM-Sigmoid Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.928	0.372	0.653	0.767	0.564
DOS	0.747	0.033	0.921	0.825	0.757
Probe	0.716	0.023	0.793	0.753	0.726
U2R	0.388	0.00	0.765	0.515	0.544
R2L	0.123	0.00	0.974	0.218	0.325
Weighted Avg.	0.745	0.174	0.798	0.718	0.618

(e) Performance Metrics Using SVM Kernels For MultiClass Classification

SVM Kernel Type	Accuracy %	TrainingTime (s)	TestingTime (s)
Linear	75.73	48.79	4.29
Polynomial	73.58	121.27	13.93
RBF	72.78	444.86	71.66
Sigmoid	74.54	177.87	16.6

ISCX 2012 Benchmark Results

Table 4.12: SVM ISCX Multi Class Confusion Matrices

(a) SVM-Linear Confusion Matrix

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>SSH</i>	<i>L2L</i>	<i>Botnet</i>
Actual Class	<i>Normal</i>	69109	4	2	181	494
	<i>DOS</i>	17	947	0	703	1
	<i>SSH</i>	2	1	3602	0	0
	<i>L2L</i>	227	1	0	9173	4
	<i>Botnet</i>	28	0	44	1	18688

(b) SVM-Polynomial Confusion Matrix

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>SSH</i>	<i>L2L</i>	<i>Botnet</i>
Actual Class	<i>Normal</i>	68931	58	115	183	503
	<i>DOS</i>	190	809	1	668	0
	<i>SSH</i>	8	0	3597	0	0
	<i>L2L</i>	561	14	114	8712	4
	<i>Botnet</i>	147	0	45	0	18569

(c) SVM-RBF Confusion Matrix

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>SSH</i>	<i>L2L</i>	<i>Botnet</i>
Actual Class	<i>Normal</i>	69203	17	2	74	494
	<i>DOS</i>	71	890	1	706	0
	<i>SSH</i>	8	0	3597	0	0
	<i>L2L</i>	257	0	0	9144	4
	<i>Botnet</i>	29	2	44	0	18686

(d) SVM-Sigmoid Confusion Matrix

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>SSH</i>	<i>L2L</i>	<i>Botnet</i>
Actual Class	<i>Normal</i>	69066	2	2	227	493
	<i>DOS</i>	95	896	1	676	0
	<i>SSH</i>	8	0	3597	0	0
	<i>L2L</i>	541	2	0	8858	4
	<i>Botnet</i>	147	0	44	0	18570

The resulting performance metrics resulting from these confusion matrices are calculated for Normal, DOS, SSH, L2L and Botnet classes and are shown in tables 4.13a - 4.13e. It can be observed that Sigmoid Kernel function gives a better detection rate compared to other kernel functions.

Table 4.13: SVM ISCX 2012 Multi Class Classification Performance Metrics

(a) SVM-Linear Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.990	0.008	0.996	0.993	0.979
DOS	0.568	0.00	0.994	0.723	0.748
SSH	0.999	0.00	0.987	0.993	0.993
L2L	0.975	0.009	0.912	0.943	0.937
Botnet	0.996	0.006	0.974	0.985	0.982
Weighted Avg.	0.983	0.007	0.984	0.983	0.972

(b) SVM-Polynomial Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.988	0.027	0.987	0.987	0.961
DOS	0.485	0.001	0.918	0.635	0.664
SSH	0.998	0.003	0.929	0.962	0.961
L2L	0.926	0.009	0.911	0.919	0.910
Botnet	0.990	0.006	0.973	0.982	0.977
Weighted Avg.	0.975	0.020	0.974	0.973	0.955

(c) SVM-RBF Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.992	0.011	0.995	0.993	0.979
DOS	0.534	0.00	0.979	0.691	0.720
SSH	0.998	0.00	0.987	0.992	0.992
L2L	0.972	0.008	0.921	0.946	0.941
Botnet	0.996	0.006	0.974	0.985	0.982
Weighted Avg.	0.983	0.009	0.984	0.982	0.972

(d) SVM-Sigmoid Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.990	0.024	0.989	0.989	0.966
DOS	0.537	0.00	0.996	0.698	0.728
SSH	0.998	0.00	0.987	0.992	0.992
L2L	0.942	0.010	0.907	0.924	0.917
Botnet	0.990	0.006	0.974	0.982	0.978
Weighted Avg.	0.978	0.018	0.979	0.977	0.961

(e) Performance Metrics Using SVM Kernels For Multi Class Classification

SVM Kernel Type	Accuracy %	Training Time (s)	Testing Time (s)
Linear	98.34	1062.17	37.03
Polynomial	97.47	7861.76	612.09
RBF	98.34	2743.12	91.55
Sigmoid	97.82	2599.07	120.14

4.4 Intrusion Detection using ELM

In this section we study the use of ELM for Binary and Multi Class classification for intrusion detection. In this research we use the following notations to describe the ELM algorithm:

N : Number of distinct training instances.

N_t : Number of distinct testing instances.

L : Number of hidden neurons.

n : Number of input neurons.

m : Number of output neurons.

$(\mathbf{x}_j, \mathbf{t}_j)$: $j=1,2,\dots,N$: arbitrary distinct (input,output) samples $\mathbf{x}_j = (x_{j1}, x_{j2}, \dots, x_{jn})^T \in \mathbb{R}^n$, expected output vectors $\mathbf{t}_j = (t_{j1}, t_{j2}, \dots, t_{jm})$. Mathematically single hidden layer feed forward networks (SLFNs) with L hidden nodes and activation function are modeled as

$$\sum_{i=1}^L \beta_i g_i(x_j) = \sum_{i=1}^L \beta_i g(w_i \cdot x_j + b_i) = \mathbf{o}_j, \quad (4.15)$$

$$j = 1, \dots, N$$

where $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T$ is the weight vector connecting the i th hidden node and input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$ is the weight vector connecting the i th hidden node and the output nodes and b_i is the threshold of the i th hidden node. The standard single hidden layer feed forward network with $L = N$ hidden neurons can approximate these N samples with zero error which means that $\sum_{j=1}^N \|\mathbf{o}_j - \mathbf{t}_j\| = 0$ i-e there exists β_i , $\mathbf{w}_i$ and b_i such that

$$\sum_{i=1}^L \beta_i g(w_i \cdot x_j + b_i) = \mathbf{t}_j, j = 1, \dots, N \quad (4.16)$$

The N equations can be written in a compact way as $\mathbf{H}\beta = \mathbf{T}$, where $\mathbf{H} = [h_{ij}]_{N \times L}$

$$\begin{bmatrix} g(w_1^T \cdot x_1 + b_1) & \cdot & \cdot & g(w_L^T \cdot x_1 + b_L) \\ \cdot & \cdot & \cdot & \cdot \\ g(w_1^T \cdot x_N + b_1) & \cdot & \cdot & g(w_L^T \cdot x_N + b_L) \end{bmatrix} \quad (4.17)$$

is the hidden layer output matrix of the neural network. The function g represents the activation function of the hidden neurons. There are many choices for the activation function which include sigmoid [80], sine [35], Gaussian [81] and linear [81]. Sigmoid function is used in this research. The function g can be written as $g(z) = 1/(1+\exp(-z))$. $\beta = [\beta_{ij}]_{L \times m}$

$$\beta = [\beta_{ij}]_{L \times m} = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdot & \beta_{1m} \\ \beta_{21} & \beta_{22} & \cdot & \beta_{2m} \\ \cdot & \cdot & \cdot & \cdot \\ \beta_{L1} & \beta_{L2} & \cdot & \beta_{Lm} \end{bmatrix} \quad (4.18)$$

are the output weights between the hidden layer and output layer and

$$\mathbf{T}_{N \times m} = \begin{bmatrix} t_{11} & t_{12} & \cdot & t_{1m} \\ t_{21} & t_{22} & \cdot & t_{2m} \\ \cdot & \cdot & \cdot & \cdot \\ t_{N1} & t_{N2} & \cdot & t_{Nm} \end{bmatrix} \quad (4.19)$$

are the output vectors. Training a single hidden layer feedforward network is equivalent to finding a least squares error solution $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$, where $\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$ is the Moore-Penrose generalized inverse of the hidden layer output matrix $\mathbf{H}$. To obtain a stable solution a positive constant C is added to the diagonal of $\mathbf{H}^T \mathbf{H}$. The output weights are then calculated as in equation 4.20 where $\mathbf{I}$ is the identity matrix.

$$\hat{\beta} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T} \quad (4.20)$$

4.4.1 Single Hidden Layer Feed Forward network Learning with Incremental L2-norm Solver on Smart Gateway

Algorithm 1: Learning Algorithm for Single Layer Network

Input : Training Set $(x_i, t_i), x_i \in \mathbf{R}^n, t_i \in \mathbf{R}^m, i = 1, \dots, N$, activation function $H(a_{ij}, b_{ij}, x_i)$, maximum number of hidden neuron node k_{max} and accepted training error ϵ .

Output: Neuron Network output weight β

- 1 Randomly assign hidden-node parameters
 - 2 $(a_{ij}, b_{ij}), a_{ij} \in \mathbf{A}, b_{ij} \in \mathbf{B}$
 - 3 Calculate the hidden-layer pre-output matrix $\mathbf{H}$
 - 4 $\text{preH} = \mathbf{XA} + \mathbf{B}, \mathbf{H} = 1/(1 + e^{-\text{preH}})$
 - 5 Calculate the output weight
 - 6 $\beta = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{T}$
 - 7 Calculate the training error
 - 8 $\text{error} = \|\mathbf{T} - \mathbf{H}\beta\|$
 - 9 **if** $(L \leq L_{max} \text{ and } e > \epsilon)$ **then**
 - 10 Increase number of hidden node
 - 11 $L = L + 1$, repeat from Step 1
 - 12 **end**
-

The basic ELM training algorithm is summarized in Algorithm 1. The highest time complexity for solving training problem is the least square problem of minimizing $\|\mathbf{T} - \mathbf{H}\beta\|$. To solve this we can use either Singular Value Decomposition (SVD), QR or Cholesky decomposition. The computational costs of each are $O(4NL^2 - \frac{4}{3}L^3)$, $O(3NL^2 - L^3)$ and $O(\frac{1}{3}L^3)$ respectively. Hence we use Cholesky Decomposition to solve the least square problem. Since its incremental and symmetric, this property reduces the overall computational cost and only utilizes half the memory required [82]. We define $\mathbf{H}_L$ to represent a matrix with L number of hidden neuron nodes ($L < N$) which can be decomposed into symmetric positive definite matrix $\mathbf{H}^T \mathbf{H}$. A positive definitive matrix is a symmetric matrix $\mathbf{H}^T \mathbf{H}$ for which all eigenvalues are positive. To check if a matrix is positive definite is to check that

Figure 4.6: A Single Layer Feed Forward Neural Network (SLFN) with Incremental L2-Norm Direct Solver

all its pivots are positive [83].

$$\mathbf{H}_L^T \mathbf{H}_L = \mathbf{Q}_L \mathbf{Q}_L^T \quad (4.21)$$

Algorithm 2: Incremental Cholesky Factorization

Input : Activation matrix $\mathbf{H}_L$, target matrix $\mathbf{T}$ and number of hidden nodes L

Output: Neuron Network output weight β

```

1 for  $L \leftarrow 2$  to  $L$  do
2   Calculate new added column
3    $\mathbf{v}_L \leftarrow \mathbf{H}_{L-1}^T h_L$ 
4    $g \leftarrow h_L^T * h_L$ 
5   Calculate updated Cholesky matrix
6    $\mathbf{z}_L \leftarrow \mathbf{Q}_{L-1}^{-1} \mathbf{v}_L$ ,  $p \leftarrow \sqrt{g - \mathbf{z}_L^T \mathbf{z}_L}$ 
7   Form new Cholesky Matrix  $\mathbf{Q}_L \leftarrow \begin{pmatrix} \mathbf{Q}_{L-1} & 0 \\ \mathbf{z}_L^T & p \end{pmatrix}$ 
8   Calculate output weight using forward and backward substitution
9    $\mathbf{Q}_L \mathbf{Q}_L^T \beta \leftarrow \mathbf{H}_L^T \mathbf{T}$ 
10 end
```

In equation 4.21 $\mathbf{Q}_L$ represents a low triangular matrix and $\mathbf{Q}_L^T$ is the

Figure 4.7: Incremental LMS Performance

transposed $\mathbf{Q}_L$ matrix.

$$\begin{aligned}\mathbf{H}_L^T \mathbf{H}_L &= [\mathbf{H}_{L-1} \ h_L]^T [\mathbf{H}_{L-1} \ h_L] \\ &= \begin{pmatrix} \mathbf{H}_{L-1}^T \mathbf{H}_{L-1} & \mathbf{v}_L \\ \mathbf{v}_L^T & g \end{pmatrix}\end{aligned}\quad (4.22)$$

where h_L is the new added column by increasing the size of L . We can express the Cholesky matrix as

$$\begin{aligned}\mathbf{Q}_L \mathbf{Q}_L^T &= \begin{pmatrix} \mathbf{Q}_{L-1} & \mathbf{0} \\ \mathbf{z}_L^T & p \end{pmatrix} \begin{pmatrix} \mathbf{Q}_{L-1}^T & \mathbf{z}_L \\ \mathbf{0} & p \end{pmatrix}\end{aligned}\quad (4.23)$$

Hence we can easily calculate z_K and scalar p for Cholesky factorization as

$$\mathbf{Q}_{L-1} \mathbf{z}_L = \mathbf{v}_L, \ p = \sqrt{g - \mathbf{z}_L^T \mathbf{z}_L}\quad (4.24)$$

Here $\mathbf{Q}_{L-1}$ represents the previous Cholesky decomposition result and $\mathbf{v}_L$ can be obtained from 4.22. This means we can continue to use the previous factorization result and update current part only. Algorithm 2 gives details on each step since $L \geq 2$ and is graphically illustrated in Figure 4.6 . We use an incremental based ELM training algorithm as shown in Algorithm

1 to perform Binary and Multi Class classification. The improvement in training time from using incremental cholesky factorization compared to the non incremental method is shown in Figure 4.7. Recall, False Positive Rate, Precision, F-Measure, MCC, overall Accuracy, training and testing times are used to compare performance over varying number of hidden neurons. Both Binary and Multi Class classification experiments are performed.

4.4.2 Binary Classification

In this section binary classification results are presented using the sigmoid kernel function for the NSL-KDD and ISCX intrusion detection benchmarks. The number of hidden neurons are 500.

NSL-KDD and ISCX 2012 Benchmark Results

Table 4.14: ELM NSL-KDD Binary Classification Performance Metrics

(a) ELM Sigmoid Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	9331	380
	<i>anomaly</i>	4368	8465

(b) ELM Sigmoid Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.961	0.340	0.681	0.797	0.629
anomaly	0.660	0.039	0.957	0.781	0.629
Weighted Avg.	0.789	0.039	0.838	0.787	0.629

Table 4.15: ELM ISCX 2012 Binary Classification Performance Metrics

(a) ELM Sigmoid Confusion Matrix

		Predicted Class	
		<i>normal</i>	<i>anomaly</i>
Actual Class	<i>normal</i>	65908	3882
	<i>anomaly</i>	3956	29483

(b) ELM Sigmoid Performance Metrics

Class	Recall	FP	Precision	F-Measure	MCC
normal	0.944	0.118	0.943	0.944	0.826
anomaly	0.881	0.056	0.884	0.882	0.826
Weighted Avg.	0.924	0.098	0.924	0.924	0.826

4.4.3 Multi Class Classification

In this section Multi Class classification results for NSL-KDD and ISCX Benchmark data sets using ELM using 500 hidden neurons are presented.

NSL-KDD Benchmark Results

Table 4.16: ELM NSL-KDD Multi Class Classification Performance Metrics

(a) ELM Sigmoid Confusion Matrix using 500 hidden neurons

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>Probe</i>	<i>U2R</i>	<i>R2L</i>
Actual Class	<i>Normal</i>	9305	345	44	0	17
	<i>DOS</i>	451	1672	293	0	5
	<i>Probe</i>	1340	509	5788	0	1
	<i>U2R</i>	67	0	0	0	0
	<i>R2L</i>	2497	182	21	0	7

(b) ELM Sigmoid Performance Metrics using 500 hidden neurons

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.958	0.339	0.681	0.796	0.627
DOS	0.691	0.051	0.617	0.652	0.608
Probe	0.757	0.024	0.941	0.839	0.779
U2R	0.00	0.00	–	–	–
R2L	0.0025	0.0011	0.233	0.005	0.012
Weighted Avg.	0.743	0.159	–	–	–

ISCX 2012 Benchmark Results

Table 4.17: ELM ISCX 2012 Multi Class Classification Performance Metrics

(a) ELM Sigmoid Confusion Matrix using 500 hidden neurons

		Predicted Class				
		<i>Normal</i>	<i>DOS</i>	<i>SSH</i>	<i>L2L</i>	<i>Botnet</i>
Actual Class	<i>Normal</i>	66074	80	1251	1247	1138
	<i>DOS</i>	591	574	66	433	4
	<i>SSH</i>	1082	0	2522	0	1
	<i>L2L</i>	2619	112	34	6604	36
	<i>Botnet</i>	140	0	0	8	18613

(b) ELM Sigmoid Performance Metrics using 500 hidden neurons

Class	Recall	FP	Precision	F-Measure	MCC
Normal	0.946	0.132	0.937	0.941	0.818
DOS	0.344	0.001	0.749	0.471	0.502
SSH	0.699	0.0135	0.651	0.674	0.662
L2L	0.702	0.017	0.796	0.746	0.724
Botnet	0.992	0.013	0.940	0.965	0.958
Weighted Avg.	0.914	0.094	0.911	0.911	0.825

It can be observed from table 4.16b and table 4.17b that ELM gives good performance metrics for Denial of Service (DOS) and Probing attacks. However it does not give good performance metrics for User to Root (U2R) and Remote to Local (R2L) attack categories. This is because the NSL-KDD dataset is highly imbalanced for User to Root (U2R) and Remote to Local (R2L) attack categories. Figure 4.8 shows that with increasing number of neurons, the classification accuracy increases up to a certain number of neurons after which it becomes constant. Figure 4.9 shows a comparison of SVM and ELM at 500 hidden neurons for both the datasets. For ISCX 2012 dataset, SVM performs better for all the different type of classes. For NSL-KDD dataset, SVM gives comparable performance to ELM for Normal, Denial of Service (DOS) and Probe Class. ELM is not able to categorize any intrusions relating to User to Root (U2R) and Remote to Local (R2L) classes.

Figure 4.8: Network Intrusion Detection System (NIDS) Classification Accuracy

Figure 4.9: Comparison Between SVM and ELM using 500 Hidden Neurons For Multi Class Classification

4.4.4 Weighted Extreme Learning Machine

It can be observed from Figure 4.3 and Figure 4.4 for NSL-KDD and ISCX 2012 datasets respectively, that the data distribution of different classes is highly imbalanced. This imbalance causes the machine learning algorithms to give more importance to classes that have a majority class distribution. To deal with this data imbalance various techniques are employed such as balancing the data distribution by sampling methods or algorithmic approaches [84] [85]. Weight adjustment is a technique proposed to deal with this class imbalance problem [86] when ELM is used as learner. Given a set

of training samples $[x_i, t_i]$, $i = 1, \dots, N$ belonging to 2 classes where t_i is +1 or -1, A weight matrix of size $N \times N$ is defined for weights associated with every training sample. The diagonal of the weight matrix holds the weights for the training samples. For instance if x_{ii} belongs to the minority class the corresponding weight will be increased so that marginal distance between the classes is maximized and the weighted cumulative error is minimized for each sample [86]. The weighting scheme used is based on the following [86] :

$$\beta = H^\dagger T = \begin{cases} H^T \left(\frac{I}{C} + W H H^T \right)^{-1} W T, & N < L \\ \left(\frac{I}{C} + H^T W H \right)^{-1} W H^T T, & N \geq L \end{cases} \quad (4.25)$$

Figure 4.10: Comparison Between SVM and Weighted ELM using 500 Hidden Neurons For NSL-KDD Benchmark Multi Class Classification

Figure 4.10 shows the sensitivity and precision for weighted ELM and results are compared against SVM and ELM for the NSL-KDD data. It can be observed that sensitivity has improved considerably, specially for User to Root (U2R) and Remote to Local (R2L) classes which had a 0 % detection rate previously when ELM was used. The sensitivity for normal and probe classes also improved where as the sensitivity for Denial of Service (DOS) class is comparable to SVM. The precision for weighted ELM is comparable for Normal and Denial of Service (DOS) classes. The precision for User to Root (U2R) is very low compared to SVM. This is because of

Figure 4.11: Comparison Between SVM and Weighted ELM using 500 Hidden Neurons For ISCX 2012 Benchmark Multi Class Classification

the large number of false positives for this class. The precision for Remote to Local (R2L) class using weighted ELM is much better compared to basic ELM which had a precision of 23.3 %. Figure 4.11 shows the results for weighted ELM for the ISCX data set. It shows that using Weighted ELM, the sensitivity improves for DOS, SSH and L2L classes compared to Basic ELM. The sensitivity drops for Normal and Botnet classes compared to Basic ELM. The precision for the DOS, SSH, L2L and Botnet classes improve where as the precision for Normal class decreases. The precision for all the classes is comparable to SVM and improves specially for the DOS classes.

Chapter 5

Distributed Machine Learning

In this section we discuss the prospects of using distributed machine learning techniques for intrusion detection on resource limited hardware such as Beagleboard-Xm. The motivation for doing so comes from the fact that embedded devices are now being used in ad hoc sensor networks where the nodes do not rely on any centralized or fixed infrastructure. Instead the nodes rely on each other for information exchange and to keep the network connected [87]. One of our main objectives was to reduce training time required for the classifier on resource limited Beagleboard-Xm hardware. Although powerful personal computers (PC's) and even graphical processing units (GPU's) exist where off line training can be performed much easily and with reduced training time, employment of such servers might not be practical for all IoT environments and situations. For example if the training model needs to be updated on line in case of on line sequential ELM where raw traffic coming to the embedded devices is processed for intrusions in real time, it is necessary to perform training on the embedded devices themselves. Data integrity and confidentiality might also be compromised if data is mishandled while transferring from the devices to other PC's.

To distribute training task on gateways, whose number is M , if the training time on m th gateway is $t_{train,m}$, then the total training time of the system depends on the gateway that consumes the maximum training time, thus systems total training time with a distributed fashion can be described as:

$$T_{train} = \max(t_{train,m}) \quad (5.1)$$

As such our objective is to reduce the training time on the distributed system:

$$\min(T_{train}, T_{train}) = \max(t_{train,m}) \quad (5.2)$$

To achieve our objectives, 2 methods are utilized. The first method is based on SVM while the second one utilizes neural networks. In the following sections each technique and experiments for validation are presented.

5.1 Intrusion Detection Using Distributed SVM

In this section we describe a framework for training our benchmark data using SVM in a distributed environment based on parallel computing paradigm. Specifically we describe algorithms which employ MPI (Message Passing Interface) protocols to distribute the computational tasks. In a similar situation Xiang [6] proposed to use ELM based on map reduce framework for detecting network intrusion attempts and to minimize training times. However map reduce framework is designed to run on commodity hardware with RAM size in range of gigabytes. Beagleboard-Xm currently only has 512 MB of RAM and hence map reduce framework cannot be used on this embedded system for IoT applications. As an alternative we use the standard MPI protocol to perform parallel computations in an attempt to reduce training time complexity.

Sequential Distributed SVM (SQ-DSVM)

Figure 5.1: Sequential Distributed SVM (SQ-DSVM)

Figure 5.1 shows the working schedule of sequential distributed SVM. Chaoyang et.al [78] proposed a parallel multi category SVM based on sequential minimum optimization-type decomposition method for support vector machines (SMO-SVM). To improve the performance using parallel computing techniques, the problem of generating a large number of sub models is decomposed and distributed among multiple processing elements (PEs). Each processing element solves a sub problem [78]. For multi classification using SVM we have $k(k-1)/2$ models where k is the number of classes. Since generation of these models is the most time consuming part it is desirable to distribute the sub models cyclically onto multiple processors and each processor performs a subtask to improve the performance. For example for multi classification if we have $k=5$ classes and 2 processors we will generate 10 sub models (binary-class classifications) which can be computed cyclically over the processors/gateways as shown in Figure 5.1a. In Figure 5.1a the first $k-1$ tasks (4) are mapped to the first processor. The next $k-2$ tasks (3) are mapped to gateway 2 and so on. In such a way the sub tasks are cyclically mapped to each processor. Similarly if we have 16 classes and 4 processors, we will generate 120 sub models according to $k(k-1)/2$. The first processor p_0 computes the first $k-1$ (15) sub models. The second processor p_1 computes the next $k-2$ (14) sub models. The third processor p_2 computes the next $k-3$ (13) sub models. The fourth processor p_3 computes the next $k-4$ (12) sub models after which the sub models are cyclically mapped back to the first processor p_0 until all the sub models have been computed. This example is illustrated in Figure 5.1b.

Most of the time in training using SVM (around 95 %) is spent on generating binary decision functions while t_{run} is negligible. The training time can be simplified as follows:

$$\sum_{m=1}^M t_{train,m} = \sum_{p=1}^{k(k-1)/2} t_p \quad (5.3)$$

5.1.1 Experiment Settings and Results for DSVM

In this section we describe our experiment using Sequential Distributed SVM for NSL-KDD training dataset of size 25192. Before verifying the effectiveness of this distributed algorithm, we test the computational capacity of PC and gateway with $25192 \times \text{Dup}$ ($\text{Dup}=1,2,\dots,5$ implies data is duplicated $\times$ the number of count). To verify the effectiveness of Sequential Distributed SVM, 3 Baselines were considered as mentioned below.

1. **Baseline 1:** Centralized SVM on PC. In this method training is performed on PC for varying number of Duplicate counts.
2. **Baseline 2:** Centralized SVM on Beagleboard-Xm. In this method training is performed on Beagleboard-Xm for varying number of Duplicate counts.
3. **Baseline 3:** Sequential Distributed SVM algorithm. In this method training is performed on Beagleboard-Xm using Sequential Distributed SVM algorithm for varying number of Duplicate counts.

Table 5.1: Training time comparison among different SVM Baselines

Dup	Time consumed (seconds)					
	SVM - PC		SVM - BB		SQ - DSVM - BB	
1	80.44	1×	828.80	10.3×	155.77	1.9 ×
2	337.43	1×	2923.34	8.7×	545.75	1.6 ×
3	746.35	1×	6239.57	8.4×	2472.31	3.3 ×
4	1285.65	1×	10770.31	8.4×	4008.34	3.1 ×
5	1976.65	1×	16467.81	8.3×	4903.05	2.5 ×

Table 5.1 shows the results for the baselines mentioned previously. Sigmoid kernel was used for training from discussion in section on intrusion detection using SVM. Table 5.1 shows the results for the baselines mentioned previously. Following observations can be made from the table:

1. Without a distributed fashion, centralized SVM on board only shows about 1/8 to 1/10 in computational ability of PC.
2. With DSVM on boards, improvement in computational efficiency in run time can be $2.7\times$ - $5.3\times$ of centralized SVM-BB and can perform now at 1/6 th to 1/3 rd of PC computational efficiency.

3. An increase in the duplication factor for data doesn't show an increase in relative training time with distributed machine learning implying that computational load is much greater than data communication.

5.2 Intrusion Detection Using Distributed Neuron Network

A distributed neuron network (DNN) based machine learning [88] can result in fast training. Figure 5.2 represents a distributed network intrusion detection system (NIDS) based on DNN with soft voting for 2 sub systems. As given in Algorithm 1, the input weights and bias are chosen randomly. This means that each Single Hidden Layer Feed Forward Neuron Network (SLFN) is an independent system for intrusion detection. Each gateway will generate posteriori class probabilities $P_j(c_i|x)$, $i = 1, 2, \dots, m$, $j = 1, 2, \dots, N_{slfn}$, where x is the received data, m is the number of classes (output neurons) and N_{slfn} are the number of sub-systems for single layer network deployed on smart gateways. During the testing process the output of SLFN will be a set of values $y_i, i = 1, 2, \dots, m$. In Basic ELM the maximum y_i is selected to represent the class i . In our case we scale the input data between $[-1, +1]$ and target labels are formed using a set of $[-1, -1, \dots, +1, \dots, -1]$ where the unique $+1$ represents its class and target label has length m . The posteriori probability is estimated as

$$P_j(c_i|\mathbf{x}) = (y_i + 1)/2, \quad j = 1, 2, \dots, N_{slfn} \quad (5.4)$$

The objective for soft voting technique is to combine the posteriori probabilities of all SLFN's sub-systems to make accurate decisions for the incoming data x . Information theory suggests to use cross entropy (Kullback-Leibler distance) criterion [89]. Using this criterion we can combine the results in either arithmetic or geometric average rule. The geometric average rule is calculated as

$$P(c_i) = \prod_{j=1}^{N_{slfn}} P_j(c_i|\mathbf{x}), \quad i = 1, 2, \dots, m \quad (5.5)$$

Figure 5.2: Distributed Neuron Network with Soft Voting

where as the arithmetic rule is calculated as

$$P(c_i) = \frac{1}{N_{slfn}} \sum_{j=1}^{N_{slfn}} P_j(c_i|\mathbf{x}), \quad i = 1, 2, \dots, m \quad (5.6)$$

where $P(c_i)$ is the posteriori probability to choose class c_i and will select the maximum posteriori $P(c_i)$ for both cases. In this research we use arithmetic average as soft-voting of each gateway since [89] indicates that geometric rule works poorly when the posteriori probability is very low. The final decision is processed at the central gateway to collect the voting value from each sub-systems on other gateways. Such soft-voting will utilize the confidence of each sub system and avoid the prerequisite that each sub-system maintains an accuracy of more than 50 % for hard-voting. For varying number of sub systems N_{slfn} , we split our training and testing data size accordingly. During the training phase each individual neural network generates a network intrusion detection system (NIDS) model in parallel. In prediction phase the posteriori probabilities of sub neural networks are combined using arithmetic soft voting technique to predict the class. Since we partition the data horizontally thereby reducing the sample size for each neural network, training and testing time is greatly reduced, while still achieving the same overall high accuracy for network intrusion detection system (NIDS). For

Binary network intrusion detection system (NIDS) to detect an attack the following conditions should be satisfied:

$$\max_index(P(c_i)) = \begin{cases} 1, & \frac{1}{N_{slfn}} \sum_{j=1}^{N_{slfn}} P_j(Normal|\mathbf{x}) \\ 2, & \frac{1}{N_{slfn}} \sum_{j=1}^{N_{slfn}} P_j(Intrusion|\mathbf{x}) \end{cases} \quad (5.7)$$

Similarly for multi class classification following conditions should be satisfied

$$\max_index(P(c_i)) = \begin{cases} 1, & \frac{1}{N_{slfn}} \sum_{j=1}^{N_{slfn}} P_j(normal|\mathbf{x}) \\ 2, & \frac{1}{N_{slfn}} \sum_{j=1}^{N_{slfn}} P_j(DOS|\mathbf{x}) \\ 3, & \frac{1}{N_{slfn}} \sum_{j=1}^{N_{slfn}} P_j(PROBE|\mathbf{x}) \\ 4, & \frac{1}{N_{slfn}} \sum_{j=1}^{N_{slfn}} P_j(U2R|\mathbf{x}) \\ 5, & \frac{1}{N_{slfn}} \sum_{j=1}^{N_{slfn}} P_j(R2L|\mathbf{x}) \end{cases} \quad (5.8)$$

5.2.1 Experiment Settings and Results for DNN

5.2.2 Distributed Neuron Network (DNN) Learning with Soft Voting on Smart Gateway Network

We performed training and testing on varying number of distributed sub systems or gateways. Table 5.2 shows a comparison of detection rate (recall), accuracy, training and testing time for various machine learning methods employed. The results show that using 500 hidden neurons, Basic ELM has better performance metrics than SVM. In particular the detection rate for anomaly class is greatly increased using basic ELM and is higher then SVM. This in turn leads to a better F-Measure and overall accuracy for basic ELM than SVM. Using distributed neuron network (DNN) it can be observed that performance metrics are comparable to basic ELM, however training and testing times are greatly reduced in proportion to the number of distributed neuron network (DNN) sub systems used. This is because each sub system operates only on a portion of the initial training data. Training and predicting a model is done in parallel.

It can be observed that for both datasets all the performance metrics

Figure 5.3: Flow Diagram For Detecting Intrusions Using DNN

stay relatively constant however the server processing time is reduced proportionally to the number of sub neural networks employed. For NSL-KDD, server processing time is reduced by $5\times$ compared to centralized NIDS and by $12\times$ compared to SVM when using 5 sub neural networks. Similarly for ISCX dataset, server processing time is reduced by $5\times$ compared to centralized NIDS and by $20\times$ when compared to SVM when using 5 sub neural networks. The reduced server processing time allows distributed neuron network (DNN) to detect intrusions in a lesser amount of time since each SLFN in distributed neuron network (DNN) has a reduced detection latency.

5.2.3 Intrusion Detection Latency

In the smart home system training can usually be performed offline and is not a major concern for intrusion detection. Even on embedded systems such as Beagleboard-Xm [72] which we use inside our HAN, training complexities

Table 5.2: Weighted Average Binary DSN Classification Performance
Metrics using 500 Hidden Neurons

Benchmark	Machine Learning Method	FP Rate %	Precision	Recall %	FMeasure %	MCC	Server Processing Time (s)
NSL-KDD	SVM	0.174	79.80	74.5	71.8	0.618	16.6
	Centralized NIDS	0.189	81.30	77.50	77.40	0.589	7.07
	DSN(2)	0.372	81.60	78.00	77.90	0.597	3.53
	DSN(3)	0.543	82.20	78.40	78.30	0.607	2.36
	DSN(4)	0.713	82.60	78.20	78.60	0.615	1.772
	DSN(5)	0.887	82.70	78.70	78.60	0.616	1.432
ISCX 2012	SVM	1.40	98.70	98.70	98.70	0.971	83.68
	Centralized NIDS	10.94	91.65	91.67	91.66	0.809	20.94
	DSN(2)	10.86	91.75	91.77	91.76	0.812	10.69
	DSN(3)	11.06	91.67	91.70	91.68	0.810	7.28
	DSN(4)	11.00	91.79	91.82	91.80	0.812	5.29
	DSN(5)	10.99	91.60	91.62	91.61	0.808	4.28

can be reduced using methods such as Cholesky decomposition [88] as previously discussed. However in real time detection latency is more important. Detection latency is defined to be the time passed from the arrival of the last packet that contains the attack until the alert generation in the network intrusion detection system (NIDS) [90]. Detection latency starts when an Intrusion Detection System (IDS) module captures the network packet until it finishes processing. Detection latency is composed of three parts [90]:

1. *Interrupt handling time*: Time spent on packet handling on the operating system kernel.
2. *Queuing delay*: Time a packet spends in a queue to be delivered for processing.
3. *Processing time*: Time spent at the network intrusion detection system (NIDS) modules.

Interrupt handling time is negligible compared to the queuing delay and network intrusion detection system (NIDS) processing time [90]. Therefore incoming packets will experience a finite delay and will be placed in a queue before being served by the processing node. This can be a major vulnerability for smart home systems since there is also a finite delay in raising an alarm to protect the system. Hence an increased queuing delay has the potential to damage the smart home system.

We can model such a queue using a $M/M/1$ queue model with a finite queue size of N packets. Packets can be assumed to arrive independently at each Intrusion Detection System (IDS) module at an arrival rate of λ

(packets/sec) and are modeled to follow a Poisson distribution. Each Intrusion Detection System (IDS) module at the Home Area Network (HAN), Neighborhood Area Network (NAN) and Wide Area Network (WAN) can be assumed to have a constant service rate of μ (packets/sec). The Intrusion Detection System (IDS) module's utilization factor is defined as

$$\rho = \frac{\lambda}{\mu} \quad (5.9)$$

The average time a packet spends in queue W_{queue} before being served can be calculated as

$$W_{queue} = \frac{\rho}{\mu(1 - \rho)} \quad (5.10)$$

where as the total delay W_{Total} is given by

$$W_{Total} = \frac{1}{\mu - \lambda} \quad (5.11)$$

For efficient intrusion detection, W_{Total} should be minimized so that intrusive packets can be detected quickly. Figure 5.4 shows the relationship between queuing delay W_{queue} , total delay W_{Total} as a function of system utilization ρ . We used a constant mean arrival rate $\lambda = 10^5$ packets per second and varied the mean service rate μ_{mean} from 10^5 packets per second to 2×10^5 packets per second which resulted in the utilization factor to vary from 5 % to 98.56 %. It can be observed that under high system utilization, the queuing delay becomes the main reason for increased total delay. To achieve a detection latency less than 1000 seconds we need a system utilization less than 82%. Hence in order to minimize detection latency, we should lower the system utilization factor ρ . To formulate the improvement in detection latency when we use distributed neuron network (DNN), we make some important assumptions:

1. The distance between the intrusion detection modules in smart grid which use distributed neuron network (DNN) is large.
2. We have a Multiple Input Multiple Output (MIMO) antenna architecture so that channels are mutually independent.

Figure 5.4: Relationship between Queuing Delay, Total Delay and System Utilization

In a system using SVM or centralized network intrusion detection system based on a single layer feed forward neural network (SLFN) the bandwidth of network traffic is constant. However using distributed neuron network (DNN) based on multiple SLFN's improves the effective bandwidth and channel capacity for the network traffic flow to each network intrusion detection system module. This is because only a portion of the original network traffic is now going through the network intrusion detection system modules. This causes server processing time to go down and as a result reduces server utilization and detection latency. Assuming an additive white Gaussian noise channel with B Hz bandwidth and S/N signal to noise ratio the Shannon channel capacity C for network intrusion detection system modules can be formulated as:

$$C = B \times X \log_2 \left(1 + \frac{S}{N} \right), \quad X = N_{slfn} \quad (5.12)$$

Here X represents the number of SLFN's used in distributed neuron network

(DNN). For centralized network intrusion detection system (NIDS) $X = 1$, while for distributed neuron network (DNN) X represents the number of SLFN sub systems used.

Table 5.3: Queuing Delay Analysis Between SVM and DNN using 500 Hidden Neurons

Machine Learning Method	Server Processing Time (sec)	Service Rate (μ_{mean})	Queue Delay (W_{queue})	Total Delay (W_{Total})	Server Utilization (ρ)
SVM	214.07	4×10^3	1187.07	1401.13	84.72
Centralized NIDS	26.68	4×10^4	3.25	29.91	10.85
DNN(2)	13.49	7×10^4	0.78	14.26	5.48
DNN(3)	8.97	11×10^4	0.34	9.30	3.66
DNN(4)	6.74	14×10^4	0.19	6.93	2.74
DNN(5)	5.33	18×10^4	0.12	5.44	2.16

Table 5.3 shows the results for detection latency when different machine learning algorithms are used. For queuing analysis, the results were simulated using a mean arrival rate of $\lambda = 4 \times 10^3$ packets per seconds. The mean server processing time is the testing time required to process data using these various algorithms. The corresponding mean service rate results are given in packets per seconds. It can be observed from the results that in case of SVM because of high server processing time the service rate is low and queue delay, total delay and server utilization factor are all very high. Centralized network intrusion detection system (NIDS) which employs a single SLFN has a very low server processing time. For centralized NIDS, this results in a $10\times$ faster service rate compared to SVM. As a result of lowered server processing time the queuing delay, total delay and server utilization factor are all reduced while service rate is increased. Using multiple DNN reduces the server processing time even further. The reason is because of improved network traffic bandwidth which is flowing through each SLFN in a DNN. The network traffic bandwidth and channel capacity are increased proportionally to the number of SLFN's in the DNN as given by equation 5.12. This causes the server processing time to decrease. The accuracy obtained in classifying intrusions are all high and remain almost constant when the number of sub neural networks inside a DNN system are increased.

5.3 Multi Level Hybrid Intrusion Detection System Models

In this section the effect of a hybrid classification based on ELM and SVM is explored. It can be observed from Figure 4.10 and Figure 4.11 that SVM and ELM have different sensitivity and precision for different classes. It will be beneficial to have a hybrid intrusion detection system in which SVM and ELM classifiers are combined to detect intrusions. In this regards, several multi level models have been proposed. Some models try to detect Denial of Service (DOS) and Probe attacks at the first level, Normal at the second level, and User to Root (U2R) and Remote to Local (R2L) at the last level [91] [92] where as others have proposed to classify Normal, Denial of Service (DOS) and Probe at the first level and Remote to Local (R2L) and User to Root (U2R) attacks at the second level [93]. A model was also proposed to classify Denial of Service (DOS) and Probe at the first level , User to Root (U2R) at the second level, Normal and Remote to Local (R2L) at the last level [94].

Figure 5.5: Multi Level Intrusion Detection System Models [10]

Figure 5.5 shows the two different types of architectures. Denial of Service (DOS) and Probing attacks are fixed at the first level of the intrusion detection system since they have minimal similarity to other types of attacks [91]. In return the User to Root (U2R) and Remote to Local (R2L) attacks are relatively similar to normal connections [91]. As such the architecture of Figure 5.5a is more suitable since User to Root (U2R) and Remote to Local (R2L) are detected first before being classified as normal

connections [10].

Figure 5.6: Classification Performance based On Multi SVM and Multi ELM

Based on Figure 5.5a, the classification performance of SVM, Multi level SVM, ELM and Multi Level ELM is shown in Figures 5.6a and Figures 5.6b respectively. It can be observed that both Multi Level SVM and Multi Level ELM perform better than stand alone SVM and ELM respectively. Based on Figure 5.5a it was proposed to use a multi level hybrid model as shown in Figure 5.7a. Figure 5.7c shows the classification performance of ELM and SVM based on this model. It can be observed that performance of ELM is almost similar for normal connections. SVM outperforms ELM for Denial of Service (DOS) and User to Root (U2R) classes where as ELM gives better performance for Remote to Local (R2L) class and performs slightly better for the probe class. Based on the results in Figure 5.7c the hybrid model in Figure 5.7b is proposed.

Figure 5.7: Hybrid Classification Based On SVM and ELM

5.3.1 Comparison of Various Machine Learning Models

Table 5.4: F-Measure Comparison between standalone, multilevel and hybrid models (%)

Machine Learning Model	Class				
	Normal	DOS	Probe	U2R	R2L
SVM	92.8	74.7	71.6	38.8	12.3
Multi Level SVM	88.3	83.6	74.4	53.3	20.1
ELM	78.1	67.5	68.4	3.7	38.7
Multi Level ELM	95.7	92.4	71.3	4.6	26.0
Hybrid SVM + ELM	88.7	88.3	81.9	53.3	44.0

Table 5.4 shows the comparison between the various machine learning models. It can be observed from the table that f-measure improves when multi level SVM is used compared to stand alone SVM except for Normal class where it decreases from 92.8 % to 88.3 %. Multi level ELM in general gives better f-measure performance compared to stand alone ELM except for R2L class where the f-measure drops from 38.7 % to 26.0 %. Using the hybrid SVM+ELM model, the f-measure for the normal class (88.7 %) is comparable

to that observed for multi level SVM (88.3 %). For the hybrid model f-measure for DOS class (88.3 %) is also comparable to multi level SVM. In case of U2R class, the hybrid model gives similar performance metric compared to multi level SVM. The hybrid model gives the best performance metric for probe and R2L class compared to other machine learning models.

5.4 Unsupervised Extreme Learning Machine

Clustering is a technique which is used to find patterns in an unlabeled dataset with many dimensions. The major advantage of using clustering algorithm is the ability to learn from and detect intrusions in the audit data without intrusion signatures [95]. There are two different approaches to anomaly detection based on clustering algorithms.

1. *Unsupervised Clustering*: In this method anomaly detection model is trained using unlabeled data that consists of both normal as well as attack traffic [95].
2. *Semi-supervised Clustering*: In this approach the model is trained using normal data only to build a profile of normal activity [95].

The premise behind the first assumption is that attack traffic will form a small percentage of the total traffic. Using this assumption attacks can be detected based on cluster sizes. Large clusters correspond to normal data and the rest of the data points, which are outliers, correspond to attacks [4] [95] [96] [97]. Based on unsupervised learning algorithms, a 2 tier intrusion detection system (IDS) architecture was proposed in [98]. In this intrusion detection system (IDS) during the first stage an unsupervised clustering algorithm classifies the payload of the packets observing one packet at a time and compressing it into a single byte of information. This classification can be added to the information decoded from the packet header (or a subset of this information), and passed on to the second tier [98]. In the second stage anomalies in each packet and in a sequence of packets are taken into consideration. Recently a clustering based method based on unsupervised ELM [12] was proposed which exhibits the learning capability and

computational efficiency of ELMs, can handle multi-class classification or multi-cluster clustering and can handle unseen data at test time directly [12]. The unsupervised ELM [12] is summarized in Algorithm 3.

Algorithm 3: The Unsupervised ELM algorithm

Input : Training instances: $\mathbf{X} \in \mathbb{R}^{N \times n_i}$, Number of hidden neurons L

Output: n_o embedded-dimensional space: $\mathbf{E} \in \mathbb{R}^{N \times n_o}$ for embedding task
Label of cluster index: $\mathbf{y} \in \mathbb{N}_+^{N \times 1}$ for clustering task

- 1 Make Laplacian graph $\mathbf{L}$ from $\mathbf{X}$ Randomly assign hidden-node parameters
- 2 $(a_{ij}, b_{ij}), a_{ij} \in \mathbf{A}, b_{ij} \in \mathbf{B}$
- 3 Calculate the hidden-layer pre-output matrix $\mathbf{H}$
- 4 $\text{preH} = \mathbf{XA} + \mathbf{B}, \mathbf{H} = 1/(1 + e^{-\text{preH}}), \mathbf{H} \in \mathbb{R}^{N \times L}$
- 5 **if** $L \leq N$ **then**
 - 6 Find the generalized eigenvectors $\mathbf{v}_2, \mathbf{v}_3, \dots, \mathbf{v}_{n_o+1}$ of the equation
 - 7 $(\mathbf{I}_k + \lambda \mathbf{H}^T \mathbf{LH})\mathbf{v} = \gamma \mathbf{H}^T \mathbf{H}\mathbf{v}$
- 8 **else**
 - 9 Find the generalized eigenvectors $\mathbf{u}_2, \mathbf{u}_3, \dots, \mathbf{u}_{n_o+1}$ of
 - 10 $(\mathbf{I}_u + \lambda \mathbf{LHH}^T)\mathbf{u} = \gamma \mathbf{H}^T \mathbf{Hu}$
- 11 **end**
- 12 Obtain the embedding matrix: $\mathbf{E} = \mathbf{H}\beta$.
- 13 Assume each row of $\mathbf{E}$ as a single point and cluster the N points into K clusters using K-means. If $\mathbf{y}$ is the clustering label for all the points then return $\mathbf{y}$ for clustering task or return $\mathbf{E}$ for embedding task

5.4.1 Experiment and Results for Unsupervised ELM

The experiment was performed using the NSL-KDD dataset. The number of hidden neurons were set at 2000. Table 5.5 shows the effect of increasing the dimension n_o for the embedding task. The clustering is done for binary classification to cluster normal and attack instances.

It can be observed that increasing embedding dimension increases the clustering accuracy up to a certain dimension after which it decreases. The

Table 5.5: Effect of Increasing Embedding Dimension on Unsupervised ELM Clustering Accuracy

Embedding Dimension n_o	US-ELM Accuracy %
2	68.0
4	82.5
6	84.9
8	87.1
10	85.5
12	86.9
14	86.1
16	79.1

Table 5.6: Comparison of Unsupervised ELM against Other Clustering Algorithms [12]

Algorithm	Accuracy %	False Positive %
US-ELM	87.10	10.10
k-Medoids	76.71	21.83
k-Means	57.81	22.95
improved k-Means	65.40	21.52
Distance-based outlier detection	80.15	21.14
EM clustering	78.06	20.74

clustering accuracy peaks at the embedding dimension of 8. Table 5.6 on the other hand shows the comparison of Unsupervised ELM against other clustering algorithms which were reported in [95]. It can be observed that US-ELM gives a better clustering classification accuracy and also gives a lower false positive rate. Having a lower false positive rate is desirable since it means normal connections are not raised as alarms or intrusions for an intrusion detection system (IDS). Based on this, unsupervised ELM algorithm which clusters normal and intrusive traffic into 2 separate clusters, multilevel intrusion detection system (IDS) based on Hybrid SVM and ELM experiment is repeated here. Figure 5.8a shows the results of ELM and SVM

(a) Classification Performance

(b) Proposed Hybrid Model

Figure 5.8: Unsupervised ELM Based Hybrid Classification

on the Multi Level model. It can be observed that SVM outperforms ELM for all the classes except for the Probe class in which ELM gives better performance. Using SVM gives an f-measure of around 99 % for the normal, Denial of Service (DOS) and Root to Local (R2L) classes.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, two machine learning algorithms SVM and ELM are studied for their application in intrusion detection. For SVM various Kernel functions are studied. It has been shown that Sigmoid kernel function is better for classification in terms of sensitivity and F-measure than other kernel functions. Basic ELM utilizing the kernel function has good detection rate for only specific type of classes. To improve the classification for these classes, Weighted ELM is utilized to improve performance. To reduce the training time complexity of SVM, Sequential Distributed SVM (SQ-DSVM) is utilized. Using ELM, Distributed Neuron Network (DNN) is developed to reduce server processing time, queuing delay and total delay. To leverage the different classification performance of SVM and ELM, a hybrid classification model is utilized in which SVM is proposed to detect Denial of Service (DOS) and User to Root (U2R) classes while ELM is proposed to detect Normal, Probe and Remote to Local (R2L) classes. Clustering algorithm based on unsupervised ELM is used to cluster normal and intrusive instances into separate clusters and the hybrid model is reevaluated. Based on the results it is proposed to use SVM to detect Normal, Denial of Service (DOS), User to Root (U2R) and Remote to Local (R2L) classes while ELM is used to detect Probe class.

6.2 Future Work

During the course of this research there are several new areas which can be explored further:

1. ***Feature Selection and Reduction:*** NSL-KDD dataset has 41 features. Further work should be performed to see which features are most relevant for intrusion detection.
2. ***Hybrid Classifier:*** Different classifiers have varying detection rates for various types of attacks. In this thesis, SVM and ELM were combined in a framework. Other classifiers such as decision trees can be added in this framework to improve the classification accuracy of different classes.
3. ***Ensemble of Classifiers:*** We can combine various classifiers in an ensemble to improve the overall classification accuracy. We can use techniques such as Boosting and Bagging to improve the classification accuracy.
4. ***Deep Machine Learning:*** Recently researchers have proposed to use Deep Machine Learning for intrusion detection. This is considerably a new area of research as there are different deep neural network architectures such as recursive neural networks, deep belief networks etc. It will be interesting to see how these classifier's perform in intrusion detection and compare to traditional classifiers such as SVM and ELM.

Bibliography

- [1] Cert advisory ca-2003-04 ms-sql server worm. [Online]. Available: <http://www.cert.org/advisories/CA-2003-04.html>
- [2] V. Jaiganesh, S. Mangayarkarasi, and P. Sumathi, “Intrusion detection systems: A survey and analysis of classification techniques,” *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, no. 4, pp. 1629–1635, 2013.
- [3] M. Ahmed, A. N. Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.
- [4] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.
- [5] N. F. Haq, A. R. Onik, M. A. K. Hridoy, M. Rafni, F. M. Shah, and D. M. Farid, “Application of machine learning approaches in intrusion detection system: A survey,” *IJARAI International Journal of Advanced Research in Artificial Intelligence*, vol. 4, no. 3, pp. 9–18, 2015.
- [6] J. Xiang, M. Westerlund, D. Sovilj, and G. Pulkkis, “Using extreme learning machine for intrusion detection in a big data environment,” in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*. ACM, 2014, pp. 73–82.
- [7] N. Komninos, E. Philippou, and A. Pitsillides, “Survey in smart grid and smart home security: issues, challenges and countermeasures,” *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 1933–1954, 2014.

- [8] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, pp. 2227–2235, 2011.
- [9] Y. Cai, S. K. Rai, and H. Yu, "Indoor positioning by distributed machine-learning based data analytics on smart gateway network," in *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*. IEEE, 2015, pp. 1–8.
- [10] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system," *Expert Systems with Applications*, vol. 67, pp. 296–303, 2017.
- [11] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.
- [12] G. Huang, S. Song, J. N. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2405–2417, 2014.
- [13] V. ENGEN, "Machine learning for network based intrusion detection," Ph.D. dissertation, Bournemouth University, 2010.
- [14] A. Lazarevic, V. Kumar, and J. Srivastava, "Intrusion detection: A survey," in *Managing Cyber Threats*. Springer, 2005, pp. 19–78.
- [15] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE network*, vol. 8, no. 3, pp. 26–41, 1994.
- [16] D. OLeary, "Intrusion-detection systems," *Journal of Information Systems*, vol. 6, no. 1, pp. 63–74, 1992.
- [17] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

- [18] K. Scarfone and P. Mell, “Guide to intrusion detection and prevention systems (idps),” *NIST special publication*, vol. 800, no. 2007, p. 94, 2007.
- [19] P.-N. Tan *et al.*, *Introduction to data mining*. Pearson Education India, 2006.
- [20] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.
- [21] V. Vapnik and C. Cortes, “Support vector networks, machine learning 20, 273-297,” 1995.
- [22] S. Mukkamala, G. Janoski, and A. Sung, “Intrusion detection using neural networks and support vector machines,” in *Neural Networks, 2002. IJCNN’02. Proceedings of the 2002 International Joint Conference on*, vol. 2. IEEE, 2002, pp. 1702–1707.
- [23] Kdd cup 1999 data. [Online]. Available: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- [24] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, “A detailed analysis of the kdd cup 99 data set,” in *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 2009.
- [25] A. H. Sung and S. Mukkamala, “Identifying important features for intrusion detection using support vector machines and neural networks,” in *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003, pp. 209–216.
- [26] S. Mukkamala, A. Sung, and B. Ribeiro, “Model selection for kernel based intrusion detection systems,” in *Adaptive and Natural Computing Algorithms*. Springer, 2005, pp. 458–461.
- [27] M. Nassar, O. Festor *et al.*, “Monitoring sip traffic using support vector machines,” in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2008, pp. 311–330.

- [28] M. Awad, L. Khan, F. Bastani, and I.-L. Yen, “An effective support vector machines (svms) performance using hierarchical clustering,” in *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*. IEEE, 2004, pp. 663–667.
- [29] W.-H. Chen, S.-H. Hsu, and H.-P. Shen, “Application of svm and ann for intrusion detection,” *Computers & Operations Research*, vol. 32, no. 10, pp. 2617–2634, 2005.
- [30] Q. Song, W. Hu, and W. Xie, “Robust support vector machine with bullet hole image classification,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 32, no. 4, pp. 440–448, 2002.
- [31] S.-J. Horng, M.-Y. Su, Y.-H. Chen, T.-W. Kao, R.-J. Chen, J.-L. Lai, and C. D. Perkasa, “A novel intrusion detection system based on hierarchical clustering and support vector machines,” *Expert systems with Applications*, vol. 38, no. 1, pp. 306–313, 2011.
- [32] C. A. Catania, F. Bromberg, and C. G. Garino, “An autonomous labeling approach to support vector machines algorithms for network traffic anomaly detection,” *Expert Systems with Applications*, vol. 39, no. 2, pp. 1822–1829, 2012.
- [33] I. Kang, M. K. Jeong, and D. Kong, “A differentiated one-class classification method with applications to intrusion detection,” *Expert Systems with Applications*, vol. 39, no. 4, pp. 3899–3905, 2012.
- [34] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: theory and applications,” *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.
- [35] C. Cheng, W. P. Tay, and G.-B. Huang, “Extreme learning machines for intrusion detection,” in *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012, pp. 1–8.

- [36] G. P. M. de Farias, A. L. de Oliveira, and G. G. Cabral, "Extreme learning machines for intrusion detection systems," in *International Conference on Neural Information Processing*. Springer, 2012, pp. 535–543.
- [37] V. Jaiganesh and P. Sumathi, "Kernelized extreme learning machine with levenberg-marquardt learning approach towards intrusion detection," *International Journal of Computer Applications*, vol. 54, no. 14, 2012.
- [38] G. Creech and F. Jiang, "The application of extreme learning machines to the network intrusion detection problem," in *NUMERICAL ANALYSIS AND APPLIED MATHEMATICS ICNAAM 2012: International Conference of Numerical Analysis and Applied Mathematics*, vol. 1479, no. 1. AIP Publishing, 2012, pp. 1506–1511.
- [39] R. Singh, H. Kumar, and R. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8609–8624, 2015.
- [40] —, "Performance analysis of an intrusion detection system using panjab university intrusion dataset," in *2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS)*. IEEE, 2015, pp. 1–6.
- [41] S. Intarasothonchun and W. Srimuang, "Improving performance of classification intrusion detection model by weighted extreme learning using behavior analysis of the attack," in *2015 International Computer Science and Engineering Conference (ICSEC)*. IEEE, 2015, pp. 1–5.
- [42] W. Srimuang and S. Intarasothonchun, "Classification model of network intrusion using weighted extreme learning machine," in *Computer science and software engineering (JCSSE), 2015 12th international joint conference on*. IEEE, 2015, pp. 190–194.

- [43] Z. Alom, V. R. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief network and extreme learning machine," *International Journal of Monitoring and Surveillance Technologies Research (IJMSTR)*, vol. 3, no. 2, pp. 35–56, 2015.
- [44] Y. Li, C. Zhang, and L. Yang, "The research of ami intrusion detection method using elm in smart grid," *International Journal of Security and Its Applications*, vol. 10, no. 5, pp. 283–296, 2016.
- [45] C. Cai, H. Pan, and G. Cheng, "Fusion of bvm and elm for anomaly detection in computer networks," in *Computer Science & Service System (CSSS), 2012 International Conference on*. IEEE, 2012, pp. 1957–1960.
- [46] V. JaiGanesh and P. SUMATHI, "An efficient intrusion detection using fast hierarchical relevance vector machine." *Journal of Theoretical and Applied Information Technology*, vol. 62, no. 1, 2014.
- [47] Z. Zhang, "Efficient computer intrusion detection method based on artificial bee colony optimized kernel extreme learning machine," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 3, pp. 1954–1959, 2014.
- [48] A. Akusok, Y. Miche, J. Hegedus, R. Nian, and A. Lendasse, "A two-stage methodology using k-nn and false-positive minimizing elm for nominal data classification," *Cognitive Computation*, vol. 6, no. 3, pp. 432–445, 2014.
- [49] L. Pan, W. Jin, and J. Wu, "A novel intrusion detection approach using multi-kernel functions," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 12, no. 4, pp. 1088–1095, 2014.
- [50] R. Jayaprakash and S. Murugappan, "Intrusion detection based on kelm with levenberg-marquardt optimization," in *Communications and Signal Processing (ICCSP), 2015 International Conference on*. IEEE, 2015, pp. 0154–0156.

- [51] M. Raj, S. Jose *et al.*, “A host based intrusion detection system using improved extreme learning machine,” *International Journal for Innovative Research in Science and Technology*, vol. 1, no. 11, pp. 327–332, 2015.
- [52] L. Jayabalan and P. Ganapathi, “Anomaly based malicious traffic identification using kernel extreme machine learning (kelm) classifier and kernel principal component analysis (kpca),” *Indian Journal of Science and Technology*, vol. 9, no. 13, 2016.
- [53] C. Ma, J. Ouyang, H.-L. Chen, and J.-C. Ji, “A novel kernel extreme learning machine algorithm based on self-adaptive artificial bee colony optimisation strategy,” *International Journal of Systems Science*, vol. 47, no. 6, pp. 1342–1357, 2016.
- [54] J. Cao, S. Kwong, R. Wang, X. Li, K. Li, and X. Kong, “Class-specific soft voting based multiple extreme learning machines ensemble,” *Neurocomputing*, vol. 149, pp. 275–284, 2015.
- [55] H. H. Bosman, G. Iacca, A. Tejada, H. J. Wörtche, and A. Liotta, “Ensembles of incremental learners to detect anomalies in ad hoc sensor networks,” *Ad Hoc Networks*, vol. 35, pp. 14–36, 2015.
- [56] Z. Ye and Y. Yu, “Network intrusion detection,” in *Proceedings of ELM-2014 Volume 2*. Springer, 2015, pp. 71–80.
- [57] S. Shukla and R. Yadav, “Voting based extreme learning machine with accuracy based ensemble pruning,” *International Journal of Computer Applications*, vol. 115, no. 22, 2015.
- [58] Y. Yu, Z. Ye, X. Zheng, and C. Rong, “An efficient cascaded method for network intrusion detection based on extreme learning machines,” *The Journal of Supercomputing*, pp. 1–16, 2016.
- [59] S. X. Wu and W. Banzhaf, “The use of computational intelligence in intrusion detection systems: A review,” *Applied Soft Computing*, vol. 10, no. 1, pp. 1–35, 2010.

- [60] J. McHugh, “Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.
- [61] Predict. [Online]. Available: www.predict.org
- [62] Caida. [Online]. Available: www.caida.org
- [63] Defcon. [Online]. Available: www.defcon.org
- [64] Adfa intrusion detection datasets. [Online]. Available: <http://seit.unsw.adfa.edu.au/staff/sites/hu/>
- [65] Nsl-kdd. [Online]. Available: <http://nsl.cs.unb.ca/NSL-KDD/>
- [66] Kyoto dataset. [Online]. Available: www.takakura.com
- [67] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.
- [68] Ics attack dataset. [Online]. Available: <http://www.ece.msstate.edu/wiki/>
- [69] K. Kendall, “A database of computer attacks for the evaluation of intrusion detection systems,” DTIC Document, Tech. Rep., 1999.
- [70] E. Nyakundi, “Using support vector machines in anomaly intrusion detection,” Ph.D. dissertation, University of Guelph, 2015.
- [71] Y. Zhang, L. Wang, W. Sun, R. C. Green II, and M. Alam, “Distributed intrusion detection system in a multi-layer network architecture of smart grids,” *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 796–808, 2011.
- [72] Beagleboard-xm. [Online]. Available: <http://beagleboard.org/beagleboard-xm>

- [73] S. Liao, A. K. Jain, and S. Z. Li, “Partial face recognition: Alignment-free approach,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 5, pp. 1193–1205, 2013.
- [74] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, and J. Abramov, “Security issues in nosql databases,” in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2011, pp. 541–547.
- [75] S. S. Kaushik and P. Deshmukh, “Detection of attacks in an intrusion detection system,” *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 2, no. 3, pp. 982–986, 2011.
- [76] E. Hernández-Pereira, J. A. Suárez-Romero, O. Fontenla-Romero, and A. Alonso-Betanzos, “Conversion methods for symbolic features: A comparison applied to an intrusion detection problem,” *Expert Systems with Applications*, vol. 36, no. 7, pp. 10 612–10 617, 2009.
- [77] ISCX 2012 dataset. [Online]. Available: <http://www.unb.ca/research/iscx/dataset/iscx-IDS-dataset.html>
- [78] C. Zhang, P. Li, A. Rajendran, and Y. Deng, “Parallel multicategory support vector machines (pmc-svm) for classifying microarray data,” in *Computer and Computational Sciences, 2006. IMSCCS’06. First International Multi-Symposiums on*, vol. 1. IEEE, 2006, pp. 110–115.
- [79] C.-C. Chang and C.-J. Lin, “Libsvm: a library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.
- [80] Q. He, T. Shang, F. Zhuang, and Z. Shi, “Parallel extreme learning machine for regression based on mapreduce,” *Neurocomputing*, vol. 102, pp. 52–58, 2013.
- [81] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, “Op-elm: optimally pruned extreme learning machine,” *Neural Networks, IEEE Transactions on*, vol. 21, no. 1, pp. 158–162, 2010.

- [82] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.
- [83] Symmetric matrices and positive definiteness. [Online]. Available: https://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/positive-definite-matrices-and-applications/symmetric-matrices-and-positive-definiteness/MIT18_06SCF11_Ses3.1sum.pdf
- [84] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
- [85] R. Akbani, S. Kwek, and N. Japkowicz, “Applying support vector machines to imbalanced datasets,” in *European conference on machine learning*. Springer, 2004, pp. 39–50.
- [86] W. Zong, G.-B. Huang, and Y. Chen, “Weighted extreme learning machine for imbalance learning,” *Neurocomputing*, vol. 101, pp. 229–242, 2013.
- [87] L. Zhou and Z. J. Haas, “Securing ad hoc networks,” *IEEE network*, vol. 13, no. 6, pp. 24–30, 1999.
- [88] H. Huang, Y. Cai, and H. Yu, “Distributed-neuron-network based machine learning on smart-gateway network towards real-time indoor data analytics,” in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 720–725.
- [89] D. J. Miller and L. Yan, “Critic-driven ensemble classification,” *Signal Processing, IEEE Transactions on*, vol. 47, no. 10, pp. 2833–2844, 1999.
- [90] N. Tsikoudis, A. Papadogiannakis, and E. P. Markatos, “Leonids: A low-latency and energy-efficient network-level intrusion detection system,” *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 1, pp. 142–155, 2016.

- [91] P. Gogoi, D. Bhattacharyya, B. Borah, and J. K. Kalita, "Mlh-ids: a multi-level hybrid intrusion detection method," *The Computer Journal*, vol. 57, no. 4, pp. 602–623, 2014.
- [92] C. Xiang, P. C. Yong, and L. S. Meng, "Design of multiple-level hybrid classifier for intrusion detection system using bayesian clustering and decision trees," *Pattern Recognition Letters*, vol. 29, no. 7, pp. 918–924, 2008.
- [93] C. Xiang, M. Chong, and H. Zhu, "Design of multiple-level tree classifiers for intrusion detection system," in *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, vol. 2. IEEE, 2004, pp. 873–878.
- [94] H. Lu and J. Xu, "Three-level hybrid intrusion detection system," in *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*. IEEE, 2009, pp. 1–4.
- [95] I. Syarif, A. Prugel-Bennett, and G. Wills, "Unsupervised clustering approach for network anomaly detection," in *International Conference on Networked Digital Technologies*. Springer, 2012, pp. 135–145.
- [96] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer networks*, vol. 51, no. 12, pp. 3448–3470, 2007.
- [97] E. Eskin, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data eleazar eskin, andrew arnold, michael prerau, leonid portnoy, sal stolfo department of computer science columbia university."
- [98] S. Zanero and S. M. Savaresi, "Unsupervised learning techniques for an intrusion detection system," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 412–419.

Appendix A

Publications List

Book Chapter:

1. Hantao Huang, **Rai Suleman Khalid** and Hao Yu*, “Distributed Machine Learning on Smart-Gateway Network Towards Real-time Indoor Data Analytics”.

Conference papers:

1. Hantao Huang, **Rai Suleman Khalid**, Hao Yu, and Fengbo Ren, “IoT Network Intrusion Detection by Online Sequential Machine Learning Accelerator,” IEEE International System-on-Chip Conference, 2017 (in preparation).
2. Hang Xu, Hantao Huang, **Rai Suleman Khalid** and Hao Yu*, “Distributed Machine Learning based Smart-grid Energy Management with Behavior Cognition”, *IEEE International Conference on Smart Grid Communications* (SmartGridComm) 2016.
3. Cai Yuehua, **Rai Suleman Khalid**, and Hao Yu*. “Indoor positioning by distributed machine-learning based data analytics on smart gateway network.” *in Indoor Positioning and Indoor Navigation (IPIN)* IEEE, 2015.