A FAST LEARNING NETWORK WITH IMPROVED PARTICLE SWARM OPTIMIZATION FOR INTRUSION DETECTION SYSTEM

MOHAMMED HASAN ALI

DOCTOR OF PHILOSOPHY

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : MOHAMMED HASAN ALI

Date of Birth : 26/03/1989

Title : A FAST LEARNING NETWORK WITH IMPROVE PARTICLE SWARM OPTIMIZATION FOR INTRUSION DETECTION SYSTEM

Academic Session : SEM2 2018/2019

I declare that this thesis is classified as:

☐ CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐ RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑ OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

_____	_____
(Student's Signature)	(Supervisor's Signature)
A9805821	Dr.Mohamad Fadli Zolkipli
New IC/Passport Number	Name of Supervisor
Date:	Date:

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Doctor of Philosophy

_____

(Supervisor's Signature)

Full Name : Dr. MOHAMAD FADLI ZOLKIPLI

Position : ASSOCIATE PROFESSOR

Date : April 2019

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : MOHAMMED HASAN ALI

ID Number : PCC15004

Date : April 2019

A FAST LEARNING NETWORK WITH IMPROVED PARTICLE SWARM OPTIMIZATION FOR INTRUSION DETECTION SYSTEM

MOHAMMED HASAN ALI

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Doctor of Philosophy

Faculty of Computer Systems and Software Engineering
UNIVERSITI MALAYSIA PAHANG

APRIL 2019

DEDICATION

Dedicated to my parents.

For their endless love, support and encouragement.

ACKNOWLEDGEMENTS

I am deeply grateful to my parents: my mother Batool Yas and my father Hasan Ali for your prayers and unlimited support. I certainly would not be where I am today without their nurture, guidance, love and care throughout my life. There are not enough words in the world to express my appreciation. Whatever I do, I will not be able to return the love, bestowal and support.

I am greatly indebted and appreciate very much to my brothers and sister for their encouragements, supports and sacrifices all the time. To my relatives and dearest beloved, a grateful thank you for your support and encouragement.

Finally, I would like to acknowledge and thank Universiti Malaysia Pahang for allowing me to conduct my research and providing the required assistance

ABSTRACT

In current days the intrusion detection systems (IDS) have several shortcomings such as high rates of false positive alerts, low detection rates of rare but dangerous attacks, and the need for a constant human intervention and tuning. Daily, there are reports of incidents such as major ex-filtration of data for the purposes of stealing identities, credit card numbers, and intellectual properties, as well as to take control of network resources. Machine learning approaches have been widely used to increase the effectiveness of intrusion detection platforms. While some machine learning techniques are effective at detecting certain types of attacks, there are no known methods that can be applied universally and achieve consistent results for multiple attack types. This situation makes the detection of cyber-based attacks on computer networks a relevant and challenging area of research. The Fast Learning Network (FLN) is one of the new machine learning algorithms that are easy to implement, computationally efficient, and with excellent learning performance characteristics. However, the internal power parameters (weight and basis) of FLN are initialized at random, causing the algorithm to be unstable. In this work, a new cooperative multi-swarm scheme called multi-swarm optimization (MRPSO) which is inspired by the human social behavior was proposed for the interaction of several PSO groups while searching for the best parameters values of PSO. The focus of this research is on the development of a model that can optimize the initial parameters of FLN based on MRPSO to obtain an optimal set of initial parameters for FLN, thus, creating an optimal FLN classifier named as MRPSO-FLN which can improve the efficacy of network intrusion on data sets that contain instances of multiple classes of attacks. These methods were tested on NSL-KDD intrusion-detection datasets and the results indicate that the proposed approaches used in the system performed well in large dataset processing. In these experiments, it was demonstrated that the FLN optimization method achieved 0.9964 which is a higher accuracy than most of the existing paradigms for classifying network intrusion detection data.

ABSTRAK

Pada masa ini, sistem pengesanan pencerobohan (IDS) mempunyai beberapa kelemahan seperti kadar tanda palsu positif yang tinggi, kadar pengesanan yang rendah serangan jarang tetapi berbahaya, dan keperluan untuk intervensi dan penalaan manusia yang berterusan. Harian, terdapat laporan kejadian seperti penapisan utama data untuk tujuan mencuri identiti, nombor kad kredit, dan sifat intelektual, serta untuk mengawal sumber rangkaian. Pendekatan pembelajaran mesin telah digunakan secara meluas untuk meningkatkan keberkesanan platform pengesanan pencerobohan. Walaupun beberapa teknik pembelajaran mesin berkesan dalam mengesan jenis serangan tertentu, tidak ada kaedah yang diketahui yang boleh digunakan secara universal dan mencapai hasil yang konsisten untuk pelbagai jenis serangan. Keadaan ini menjadikan pengesanan serangan berasaskan siber pada rangkaian komputer yang relevan dan mencabar bidang penyelidikan. Rangkaian Pembelajaran Cepat (FLN) adalah salah satu daripada algoritma pembelajaran mesin baru yang mudah dilaksanakan, berkomputeran dengan baik, dan dengan ciri prestasi pembelajaran yang cemerlang. Walau bagaimanapun, parameter kuasa dalaman (berat dan asas) FLN diisytiharkan secara rawak, menyebabkan algoritma tidak stabil. Dalam usaha ini, satu skim berbilang kooperatif baru yang dikenali sebagai pengoptimuman multi-swarm (MRPSO) yang diilhamkan oleh tingkah laku sosial manusia dicadangkan untuk interaksi beberapa kumpulan PSO sambil mencari nilai parameter terbaik PSO. Tumpuan penyelidikan ini adalah mengenai pembangunan model yang dapat mengoptimumkan parameter awal FLN berdasarkan MRPSO untuk mendapatkan set parameter awal optimum untuk FLN, dengan itu, mewujudkan pengelas FLN optimum yang dinamakan MRPSO-FLN yang boleh meningkatkan keberkesanan intrusi rangkaian pada set data yang mengandungi contoh pelbagai kelas serangan. Kaedah-kaedah ini telah diuji pada dataset pengesan pencerobohan NSL-KDD dan hasilnya menunjukkan bahawa pendekatan yang dicadangkan yang digunakan dalam sistem dilakukan dengan baik dalam pemprosesan dataset yang besar. Dalam eksperimen ini, ia menunjukkan bahawa kaedah pengoptimuman FLN mencapai 0.9964 yang merupakan ketepatan yang lebih tinggi daripada kebanyakan paradigma sedia ada untuk mengelaskan data pengesanan pencerobohan rangkaian.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IDS	Intrusion Detection System
ELM	Extreme Learning Machine
SVM	Support Vector Machine
MLP	Multilayer Perceptron
PSO	Particle Swarm Optimization
PSO-FLN	Particle Swarm Optimization-Fast Learning Network
m	Number of Neurons
Itr	Number of Iterations
GA	Genetic Algorithm
DR	Detection Rate
FAR	False Alarm Rate
PSOPC	Particle Swarm Optimizer with Passive Congregation
KNN	K Nearest Neighbours
BP	Back Propagation
SFNN	Single Layer Feedforward Neural Network
MFNN	Multilayer Feedforward Neural Network
DPFNN	Double Parallel Forward Neural Network
SLFN	Single Hidden Layer Feedforward Neural Network
ANN	Artificial Neural Network
SRC	Sparse Representation Classification
FNN	Feedforward Neural Networks
NIDS	Network Intrusion Detection System
ML	Machine Learning
AI	Artificial Intelligence
BPNN	Back Propagation Neural Network
KDD	Knowledge Discovery in Database

CHAPTER 1

INTRODUCTION

1.1 Overview

This chapter presents theoretical model and motivations for the proposed research. It discusses the problem statement, states the objectives and describes briefly the scope for the proposed research. The chapter is divided into seven sections. Section 1.2 introduces a brief background of the proposed work with problem statement background. Section 1.3 summarizes the problem statement which includes IDS problems and proposed machine learning problem. Section 1.4 represents the research questions which this work tries to answer. Section 1.5 highlights the main research goal and objectives of this work. Section 1.6 provides a scope of the proposed model. Section 1.7 highlights the thesis organization.

1.2 Background

People have over the years depended on technology and computer networks for their daily activities, such as messaging, shopping, and marketing. These networks are constantly exposed to several online threats and for this reason, their integrity and availability ought to be protected against violation and intrusion. The experienced intruders, terrorist organizations, and rival corporations have the motive and capability to carry out developed attacks against computer systems (Choo, 2011), and this makes systems security an important issue for many researchers. Network intrusion or attack is a situation where an unauthorized user (attacker) tries to exploit the vulnerability of a system to gain access to network resources and cause a disruption in the normal operation of the network. If the attacker gains access to the network resources, it can

have an illegitimate access to the network data, can modify or corrupt the system, and can hijack the network or alter its behaviour.

In 2015, the U.S. Director of NSA, Adm. Michael Rogers, in the House Intelligence Committee warned of an impending major security attack in the U.S. in the next decade. In his words, "It's only a matter of the 'when,' not 'if,' that we are going to see something dramatic." Several state-backed hackers have continuously launched attacks on industrial-control systems that manage vital infrastructures such as nuclear power, power grid, transportation systems, and air-traffic control.

The NSA director also opined that based on his own assessment, the U.S. may fall into these attacks (Fossaceca, 2015). Furthermore, security is one of the biggest obstacles that hamper the widespread adoption of technology (Freire & Inácio, 2014). There are various studies in the literature discussing the security issues (Latif et al., 2014; Modi & A, 2013; Rong et al., 2012). Figure 1.1 shows that the direct spending on cybersecurity solutions like firewalls and IDSs is rising steadily based on the Gross Domestic Product (GDP) percentage of the U.S. from 2009 to 2017. Several tools and techniques have been proposed for the safety of the technology environment. The intrusion detection system (IDS) is one of the powerful software or hardware that is used to monitor computer network for the detection of normal or abnormal behaviours (Vasilomanolakis, 2015). An IDS monitors a network for signs of invasion which could manifest in abnormal system behaviours or violation of network security policies.

Therefore, the goal of the IDS is to detect any form of compromise in the integrity, confidentiality, and availability of a network. Since Denning introduced the concept of detecting intrusion detection (Denning, 1987), many efforts have been channeled on the ability for network monitoring tools to automatically detect network intrusion. An IDS is a system that automatically monitors network or system activities, analyzes them for any sign of intrusion, and often prevents unauthorized network access (Scarfone et al., 2007). The IDS observe the activities of a network and determine the nature of the observed activities (whether malicious or normal) based on the integrity, confidentiality and resource availability of the system (Toosi, 2007). There are several limitations of the conventional IDS (Al-Yaseen et al., 2017; Shah &

Issac, 2018), such as high rates false alarms, lack of continuous adaptation to changing malicious behaviours, and highly uneven data distribution.

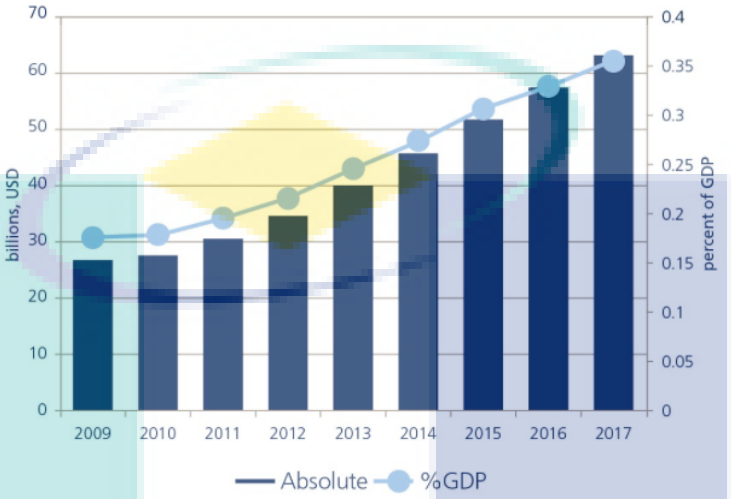


Figure 1.1 Cybersecurity spending in United States, percent of GDP and USD billions, 2009-2017, Source: TIA's 2010-2017 ICT Market review and forecast, available at: test.tiaonline.org/resources/market-forecast

The traditional IDS still suffer from several limitations as they have failed to fully protect network systems from the increasingly proliferating attacks. Most systems are built based on the traditional techniques and they suffer from false negative detections and high false alarm rates. However, several studies have reported the application of several machine learning techniques to IDS for improving their detection rates (Shah & Issac, 2018; Tsai et al., 2009).

The problem addressed by this research is that current-day network IDS suffer from several issues, including high rates of false alarms, as well as missed detections of real attacks (Baiad et al., 2016; Y. Huang et al., 2016; Shah & Issac, 2018), and therefore, requires a significant level of adjustment and tuning by human operators. This has resulted in systems that are not reliable or effective in real-world network operational environments. The current IDSs, although have been improving over time, are still not able to effectively and efficiently deal with the current threat landscape. As explained in the preceding sections, network intrusion-detection system that suffers from an excessive number of false alarms can lead network operators to ignore alerts.

There are numerous recent examples where serious attacks have been missed and classified as a normal activity, such as what happened in Ukraine which resulted in power outages, affecting approximately 225,000 customers for several hours (Liang et al., 2017). However, network operators prefer IDS that do not require human tuning, intervention, or that requires a great deal of domain knowledge to operate (Hofmeyr, 2005).

As new attacks are invented regularly, it is not sufficient to rely on classical signature-based IDS. This has motivated studies that focus on the anomaly-based machine learning for IDS (Fossaceca et al., 2015). The performance of IDS is improved by the incorporation of machine learning (ML), ML algorithms can theoretically achieve optimum performance, i.e. it can regulate the rate of false alarms and improve the detection accuracy (Snoek et al., 2012). Many ML algorithms have been proposed as IDS (Shah & Issac, 2018), and their preference (as in Extreme Learning Machine (ELM)-based IDS) is due to their proven superiority over the classical Support Vector Machine (SVM)-based IDS in different perspectives, such as over-fitting avoidance and computational cost (Guang-Bin et al., 2004; Fossaceca, 2015; Singh et al., 2015; Zamani & Movahedi, 2013), nevertheless, these IDS-based methods and algorithms still face several limitations in terms of the accuracy of detection due to the random selection of their parameters.

To enhance the performance of ML algorithms, certain characteristics have been used to hybridize them with optimization algorithms. First, each function evaluation may require a different amount of time (it takes less time to train a neural network with 10 hidden compared to a network with 1000 hidden units) (Snoek et al., 2012). Second, the impact of random selection on the main parameters(weights and biases) ought to be reduced (Zeng et al., 2017). Moreover, in terms of performance and improved detection accuracy, the IDS-based hybrid models (machine learning and optimization algorithms) have shown better results compared to single algorithms (Aslahi-Shahri et al., 2016; Goodarzi et al., 2014). In this work, the standard Particle Swarm Optimization (PSO) and Multi Swarm-based Particle Swarm Optimization (MRPSO) were used to select the main parameters and to reduce the impact of randomization on the IDS-based Fast Learning Network algorithm (FLN) as a core algorithm for the IDS proposed models.

1.3 Problem Statement

The handling of cyber threats and detection of intrusions are challenging areas in the field of information assurance. Network intruders deploy several mechanisms to evade detection techniques (Wang et al., 2018). To detect network attacks, the IDS may be equipped with ML algorithms to achieve better accuracy performance. The accuracy of IDSs has been enhanced using several supervised and unsupervised ML approaches (Lei Zhang & Zhang, 2017).There are several works in the literature on the enhancement of the IDSs accuracy based on machine learning models, such as Support Vector Machines (SVM) (Aburomman & Ibne Reaz, 2017), Artificial Neural Networks (ANNs)(Hodo et al., 2016), Backpropagation (BP) (Tran et al., 2017), Naïve Baysian Classifier (NBC) (Mukherjee & Sharma, 2012), K-nearest neighbour (KNN) (Lin et al., 2015), and Extreme Learning Machine (ELM) (Fossaceca et al., 2015; Singh et al., 2015).

The FLN algorithm is an enhanced version of ANNs which has been recently developed by Li et al. (2014) and applied to several regression and classification problems (Li et al., 2017). FLN is a double-parallel forward neural network (DPFNN) (Wang et al., 2011), made up of a parallel multilayer and SLFN networks. The output of the DPFNN nodes can receive the re-codified external information through the hidden nodes, and can also directly receive the external information through the input nodes. However, some limitations still persist, such as the randomly selection of the hidden biases and assigned input weights which may not represent the optimum performance parameters. The optimization algorithms (i.e. metaheuristics) have been utilized for enhancing the machine learning models in terms of maximizing their accuracy and minimizing their error rate for intrusion detection system. Metaheuristics have been implemented as training algorithms for various versions of ANNs, such as FFNN (Mirjalili et al., 2012), BP (Huang et al., 2015), and ELM (Zeng et al., 2017).

The training algorithms are algorithms that can search the optimal values for weights and biases. As mentioned earlier, the FLN has a drawback of the initial weight values and biases (Niu et al., 2017). They are initialized randomly and this may affect their prediction rate. Therefore, finding the optimal values of these two types of parameters (weights and biases) is an optimization problem that needs to be resolved. To overcome these issues, a multi-swarm approach called Meeting Room Approach is

proposed for tuning the PSO parameters called (MRPSO). The proposed algorithm is used as a training algorithm for enhancing the detection rate and accuracy of FLN-based network IDS named (MRPSO-FLN). In summary, the development of a machine learning-based IDS is highly promising. This work aims at filling this research gap.

1.4 Research Questions

This section provides the questions that have been framed to set the direction of this research.

RQ 1. What are the best PSO (c_1, c_2, w) parameters values to be considered for testing?
RQ 2. How effective is the basic FLN algorithm and, how to choice best weights and biases of FLN-based IDS?
RQ 3. How can the proposed models (FLN, PSO-FLN, MRPSO-FLN) accuracy be evaluated?

Given these prospects, this study presents the design and implementation of an enhanced meta-heuristic (PSO) with FLN based on IDS.

1.5 Goal and Objectives

The main goal of this study is to enhance the accuracy rate of network IDS using FLN. In addition, this work aims to build an optimized FLN model which contains a training algorithm based on metaheuristics. Reaching this model is achieved through the following objectives:

i- To propose a self-parameters tuning technique for the Particle Swarm Optimization (PSO) algorithm using a multi-swarm approach.

ii- To design a new training algorithm for Fast Network Learning (FLN) based on the proposed PSO algorithm for IDSs.

iii- To evaluate and test the prediction accuracy of proposed models (FLN, PSO-FLN, MRPSO-FLN) based on IDS dataset NSL-KDD.

1.6 Scope

This study developed an IDS and addressed the problem of IDS from the classification perspective. The focus is to develop a novel machine learning system which can serve as a highly-qualified machine learning system for intrusion systems to provide better accuracy. The system will consider learning on NSL- KDD datasets. The datasets that are selected were collected from networks with different cases of attacks and captures a wide range of network and protocol features. The system was tested on off-line mode based on the considered dataset. An online evaluation of the system was not included in the scope of this study. The evaluation was based on the main accuracy of models, while sub-attacks were not considered. The focus is on increasing the accuracy of the classifier in terms of distinguishing between attacks and non-attacks. The system can be generalized to any classification application.

1.7 Thesis Organization

This thesis is organized into 5 chapters. Chapter 1 presented an overview of security background and IDS based on ML algorithms. Then, an overview of machine learning was introduced in line with the proposed ID-based fast learning network. Finally, the problem statements, research scope, study aims, and objectives were highlighted.

In chapter 2, the classification and techniques of the IDS were reviewed, followed by an overview of the ML algorithms based on IDS and their classification. This chapter also provided an overview of the mathematical notation for Fast learning network and Particle swarm optimization. Furthermore, the existing work was analyzed to serve as a justification for the need to develop FLN.

Chapter 3 detailed the design and implementation of the basic FLN, FLN-based optimization algorithms and a new multi-swarm approach (Meeting Room Approach) based PSO for parameters tuning. This chapter discussed the issues relating to the reduction of the impact of randomly selected parameters on FLN.

In chapter 4, a detailed account of FLN evaluation was presented. Here, the performance of the FLNs was evaluated based on several evaluation measurements (Precision, Recall, F_measure, G_mean, detection rate, false alarm rate and accuracy).

The evaluation was executed using NSL-KDD datasets. Additionally, the performance of FLN was compared to other strategies.

Finally, chapter 5 presented a conclusion of the study, listing the achievements and study contributions. The conclusions were drawn from the outcome and significance of this study. Furthermore, directions for future works were also provided in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

In this chapter, a literature review of past research is presented covering the major areas relevant to the research problem. The chapter is divided into eight sections. Section 2.2 introduces briefly of the infection vectors of networks and devices and some of the popular tools of security tools. Section 2.3 gives an introduction about the IDS, the types of IDS as signature and anomaly, as well as provides a review of techniques and algorithms adopting machine learning approach. Moreover, this part highlights the data set contents, and its limitations. Section 2.4 highlights the machine learning algorithm based intrusion detection and classified them such as single and hybrid models. Section 2.5 provides an overview of neural network applications, and the new versions of algorithms that are developed based on artificial neural networks along with its limitations. Moreover, the section also explains the enhanced techniques based on artificial neural network. Section 2.6 gives an overview of Fast Learning Network algorithm, this part also contains explained the basic extreme learning machine and its limitations. Section.2.7 discusses the particle swarm optimization, which includes the standard PSO and multi swarm approach for PSO along with artificial neural network based on hybrid with PSO. Section 2.8 represents the related work with similar the main idea on the development of IDSs based on machine learning. Finally, section 2.9 provides the summary of the chapter.

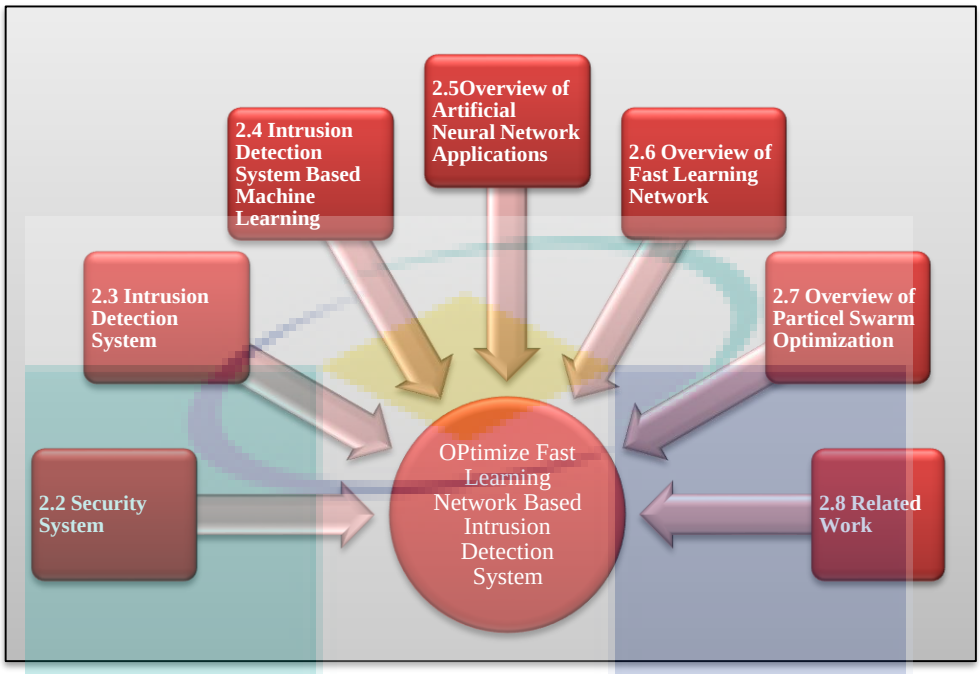


Figure 2.1 Literature Review Framework

2.2 Security System

Over the years, the Internet and computer systems have been exposed to numerous security challenges due to the increased rate of their usage. Unauthorized access to computers or information systems could result in serious security breaches and violation of security policies, availability, integrity and confidentiality (Von Solms & Van Niekerk, 2013). Computer networks have also been subjected to various traditional attacks like routing information protocol attack, DNS poisoning, address resolution protocol spoofing, IP spoofing, flooding, distributed denial of service (DDoS), and denial of service (DoS) (Brooks 2009). On the other hand, there are many systems and models which are used as tools and systems in the network environment.

The front access points of systems are protected by Firewall and treated as the first line of defense. These Firewalls are deployed to either deny or allow ports, protocols or IP addresses (Borisaniya, Patel, et al., 2013a). The Firewalls are a good option for the prevention of external attacks but does not protect from internal attacks (Borisaniya, Patel, et al., 2013a). An efficient IDS must be integrated into the networks to handle these attacks.

2.2.1 Security Tools

Technological advancements in the present world have made connectivity easier than ever. A large amount of information (personal, military, government, and commercial) are hosted on networking infrastructures worldwide. The security of network infrastructures is attracting great research interest due to the huge number of intellectual properties which can be easily acquired through the internet. The society has become over-reliant on technology as people depend on computer systems for their daily information and entertainment (Bhavya Daya, 2013). There is a need to protect the availability and integrity of these systems against several threats. experienced hackers, terrorists, foreign governments, and rival corporations can lunch sophisticated attacks against computer system (Choo, 2011). Hence, information security has become significantly important and ought to be protected against threats for the safety and economic well-being of the society.

The security of information systems has attracted several research attention due to the rapid development and wide use of electronic data processing techniques executed through wired and wireless networks, web application, and the Internet. Furthermore, the activities of numerous terrorist groups also justify the need for securing information systems. The current approaches for securing information systems are through firewalls, encryption, authentication, IDS, and prevention systems, and other hardware and software solutions (Liao et al., 2013). Most of these systems facing several limitations (Patel et al., 2013), such as Firewalls are used to protect the front access point of systems (first line of defense). They are used to either allow or deny protocols, IP addresses or ports (Naru et al. 2010). Firewalls have become paramount to the security of network infrastructures. In the network systems, firewalls are used to protect networks from external attacks by filtering and managing the Internet traffic. Figure 2.2 depicts the basic approach to the installation of firewalls (installed at the servers' entry point), showing how they form the first line of network defense.

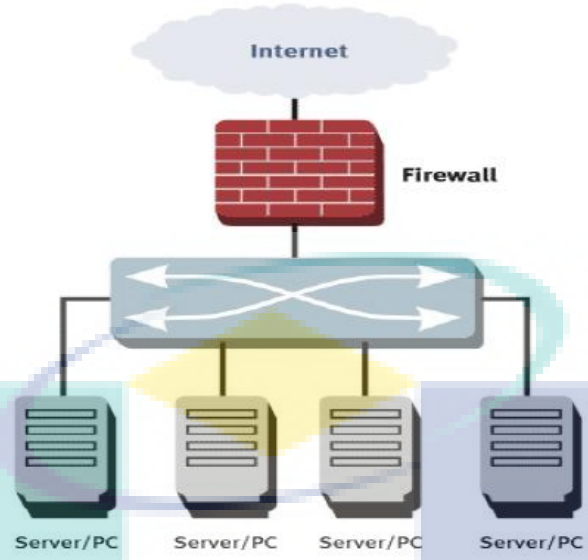


Figure 2.2 Basic firewall installation Modi et al,(2013)

Firewalls do not offer a complete analysis of all the data packets in a network traffic. They only check the packets at the network boundaries (Modi et al., 2013b). The traditional firewalls cannot detect internal attacks, and the detection of few DoS or DDoS attacks using traditional firewalls is complicated. Intelligent intrusion detection system is a dynamic defensive system that is capable of adapting to dynamically changing traffic pattern and is present throughout the network rather than only at its boundaries, thus helping to catch all types of attacks(Sivatha et al., 2012). Moreover, IDS performs the following functions (Kaur et al., 2014):

- Monitors and analyzes the activity of the system users.
- Audits the vulnerabilities and configuration of the system.
- Checks the critical system and data file integrity.
- Statistically analyzes the pattern of activities of the network based on the known attack signatures.
- Analyses of abnormal system activities.

2.3 Intrusion Detection System

Attacks from external sources are referred to as outsider attacks, while insider attacks comprise all unauthorized attempts by the internal users to gain access to unauthorized privileges. Intrusion detection involves the monitoring of computer

networks for unauthorized access, illegal activities, or file modification (Modi et al., 2013; Whitman & Mattord, 2012). Figure 2.3 depicts the use of IDS to monitor a network infrastructure. The monitored network was simultaneously connected to a network administrator who sends alarms when detected. Network attacks are mostly launched in specified groups known as incidents, and even though many incidents are dangerous in nature, most are not harmful. For instance, the address of a website may be wrongly typed and accidentally attempts to establish an unauthorized connection to a different system. Intrusion detection (ID) refers to the identification of an ongoing intrusion on a system. Some of the issues experienced in the ID studies border on data collection/reduction, behavior classification, reporting, and response (Frank, 1994). Most ID systems focus on behavior classification and data reduction. Data reduction involves the analysis of a set of data to identify the important components and reduce the processing time, communication overhead, and storage requirements. Behavior classification involves the process of identifying attacks and intruders

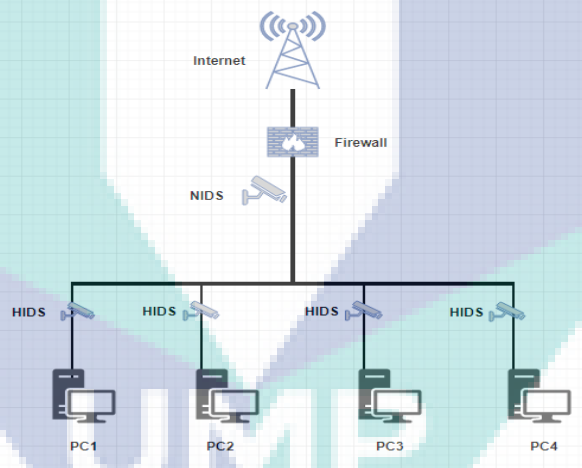


Figure 2.3 Intrusion Detection System Infrastructures
Source: Patel, et al., (2013)

Another important factor that affects the effectiveness of IDS is the quality of the deployed feature construction and selection algorithms. The goal of improving the overall effectiveness of IDS can be achieved through a guided reduction in the number of relevant features without compromising the classification accuracy of the system (Franke et al. 2012). Several techniques, such as AI and ML algorithms have been

used to achieve these aims. They work as a hybrid technique that combines two or more techniques. They are advantageous as each of the combined techniques compliment the benefits of each order (Modi et al., 2013).

The IDS monitor user activities and track network traffic to determine the nature of ongoing activities in the network. In the presence of malicious activities, the IDS generate an alarm. Several techniques, such as anomalies or signatures of attack are used by the detection system for the detection of attacks, and these techniques determine the effectiveness of an IDS (Gohil, 2015). IDS can be grouped into 2 based on their placement in the network; one group is the host-based IDS (HIDS) and the other is the network-based IDS (NIDS), as shown in Figure 2.3 (Kwon et al., 2017). IDS can further be classified based on how the execute intrusion detection; they can be classified into misuse or signature-based and anomaly detection systems. In the next sections, an overview each group is provided.

2.3.1 Classifications of Intrusion Detection System

2.3.1.1 Host Based Intrusion Detection (HIDS)

The host-based IDS (HIDS) is a group of intrusion detection system which focuses on monitoring and analyzing information sourced from a specific host system (Modi et al., 2013). The HIDS detects intrusion on its host machine by analyzing information collected from the system, such as network events, the file system used, system calls, etc. The HIDS depends on the characteristics of a system to observe any modification in the kernel and host file system. Figure 2.4 shows some host machines with installed HIDS. The figure portrays how each host or server with IDS is being monitored. Each HIDS monitors and detects unauthorized invasion of its host machine.

There are two components of IDS service – analysis and alert system. Data is captured by the event auditor from various sources such as system logs, and based on the sourced data, the IDS can detect any form of intrusion using either behavior or knowledge-based techniques. The IDS detect known to attack using the knowledge-based technique but use the behavior-based technique to detect unknown attacks. To detect unknown invasion, ANN can be used with this approach (Shah & Trivedi,

2012). In the presence of an active intrusion, the IDS sends alert to all the other nodes using the alert system (Modi et al., 2013). This is an efficient approach for the detection of known attacks via the knowledge-based technique, as well as unknown attacks by applying feedforward ANN.

2.3.1.2 Network Based Intrusion Detection System (NIDS)

A network-based IDS (NIDS) is a form of IDS which monitors network activities to detect malicious activities such as port scans, DoS attacks, or even attempted network intrusion (Modi, 2013). The system collects information from the network and compared it with known attack signatures. The NIDS has a stronger detection mechanism as it compares the current traffic behavior with an already established attack signature. The NIDS mostly detects intrusion by monitoring the IP and transport layer headers of the data packet. The NIDS uses both anomaly and signature-based techniques for intrusion detection. The NIDS are rarely visible inside their host. In an encrypted network traffic, there is no need to decrypt the traffic prior to analysis (Modi et al. 2013). Figure 2.4 shows the positioning of NIDS and their demands in a typical network. The figure also shows how NIDS focus on monitoring a group of servers which makes the difference from the HIDS. The NIDS is positioned between the network hosts and the installed firewall (Kumar, 2015). They are superior to the HIDS because HIDS can only protect one system while NIDS can protect all the systems connected to the network.

2.3.2 Intrusion Detection System Techniques

2.3.2.1 Signature Based On IDS

The signature-based IDs detects new attacks by defining a set of signatures or previous knowledge base for deciding the pattern of a given intrusion (Hubballi & Suryanarayanan, 2014). Consequently, the signature-based IDS can reach a high level of accuracy and low false positive alarm rates when identifying subtle intrusions (Brown et al., 2002). If these systems are poorly configured, the can be affected by a slight variation in the nature of known attacks. They are efficient for detecting known attacks but may fail to detect variants of a known attack (Hubballi & Suryanarayanan, 2014; Kevric et al., 2017). One reason for using signature-based IDS is the ease of

updating and maintaining their preconfigured rules. These signatures contain several traffic identification elements (Modi, Patel et al. 2013), but cannot be used in the traditional networks for the detection of unknown attacks. This limitation impacts the systems' performance because any variation in the attack signatures results in performance compromise.

2.3.2.2 Anomaly Based On IDS

The anomaly-based IDS are developed for uncovering the patterns of normal behaviours. The system establishes a benchmark of normal usage behaviours and considers any deviation from the set benchmark as an intrusion (Thatte et al., 2011). Intrusions considered to be an anomaly can vary, but normally, any incident that occurs with a frequency of more than or less than 2SD from the statistical norm is a suspect (Bringas and Penya, 2009). The anomaly-based IDS do not operate based on a database of previously known signatures; therefore, they can detect unknown intrusions and insider abuses (Maggi et al. 2009). Several studied have proposed the classification of anomaly-based IDS into static and dynamic anomaly detection. The static classification presumes that the behaviour of monitored targets cannot change (Wu and Banzhaf 2010), while the dynamic anomaly detection presumes that they extract patterns from behavioural habits of the end users or use the history of networks. The anomaly-based IDS can be generally categorized into three categories based on the processing involved (Hoang, Hu et al. 2009):

1. **Statistical anomaly detection methods:** These systems operate on two profiles: a normal profile which is built during the training phase, and the current profile built during the detection phase. The system monitors network activities like CPU usage and the number of TCP connections in terms of statistical distribution. These two profiles are compared during an active connection to facilitate the identification of anomalies when there is a statistically significant difference between the profiles. This method suffers from the difficulty of determining what constitutes a meaningful activity.
2. **Data-mining-based methods:** These methods automate the finding of meaningful activities/features. They comprised of classification-based ID, associate rule discovery, and clustering/outlier detection. In general, they

require a lot of computational effort and often produces high rates of false alarm.

3. **Machine learning-based methods:** These are also known as call-based sequence analysis. They are one of the commonly used ID techniques. The ANN is a notable example of this group. Based on the analysis of the recent trend in anomaly detection studies, several ML methods have been proposed and reported to have a high detection rate and at the same time, keeping a low rate of false alarm (Kevric et al., 2017).

The anomaly detection techniques can be deployed for the detection of new attacks at different levels in network structure. Also several system events do occur which makes it difficult to control or monitor them which make researchers prefer using the anomaly detection systems (Modi et al. 2013). Several instances have shown the use of anomaly detection techniques for the detection of intrusions at different datasets (Garfinkel and Rosenblum 2003; Dastjerdi et al. 2009; Vieira and Schuler 2010). Because of these advantages of anomaly IDS, this work will apply the new model based on the anomaly detection. Table 2.1 shows a brief comparison of the anomaly and signature-based detection techniques based on our literature review.

Table 2-1 comparison between Anomaly and signature detection

Aspects	Anomaly Detection	Signature
Characteristics	Uses the deviation from normal usage patterns to identify intrusions.	Uses the patterns of known attacks (signatures) to identify intrusions.
Drawbacks	-Has to study sequential interrelation between transactions - False positives.	- Known attacks have to be hand-coded - Unable to detect new attacks - Need signatures update

2.3.3 Intrusion Detection System Challenges

Studies have been ongoing on new systems for an automatic detection of abnormal system usages. Moreover, Denning reported the development of an intrusion detecting model, which he suggested as a framework for a general-purpose IDS (Denning, 1987). Since then, experts have developed and applied several algorithms for automating the process of network intrusion detection. They have also continually pursued more accurate, faster and scalable methods for this purpose. With the arrival of the "IoT" era, it is expected that the number of connected devices would exceed 26 billion by the year 2020 (Gartner,2013). With this trend, the type and number of

cybersecurity issues are also expected to increase. Figure 2.4 shows some of the common challenges of IDS.

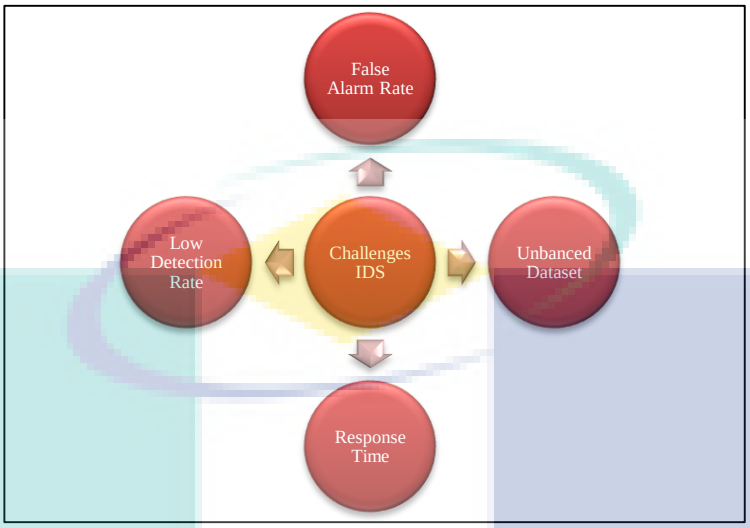


Figure 2.4 IDS challenges

Some researchers have recently advocated for more categories of IDS. Liao et al. (2013) for instance, claimed that IDS should be further categorized into 5 sub-categories which may belong to any of the aforementioned classes. The suggested sub-classes are pattern-based, rule-based, statistics-based, state-based, and heuristic-based IDS. Meanwhile, such a classification could result in confusion due to the number of similarities between the strength of the individual techniques, as well as the lack of clear criteria that distinguishes one technique from the other. The signature or rule-based IDS generally have rates of false positive rates but suffer from an inability to capture new types of attacks (Fossaceca et al., 2015). Systems that are built to detect anomalies should have a high rate of false positive alarm detection. The ID systems that are based on stateful protocol analysis present varying detection performances based on the level of their profile definition (Ghorbani et al., 2010). A major challenge of this approach is keeping an up-to-date profile as new protocols evolve over time.

As earlier discussed, this study is focused on the development of an anomaly-based IDS with a good accuracy and a minimal false positive detection. Many studies have been performed on false alarm reduction in IDS. Pietraszek estimated that about 99% of ID alerts do not involve cyber-security issues due to the observed slight

differences between normal and malicious activities (Moradi & Zulkernine, 2004). The other challenged pointed by Pietraszek include the development of accurate signatures that can capture attack behavior but not triggered during legal operations since some activities may be allowed under certain conditions but considered as suspicious at others. The "Adaptive Learner for Alert Classification" (ALAC) approach proposed by Pietraszek uses ML techniques, coupled with a human-based observatory training to adaptively learn the implicit classification rules. Due to the involvement of human factor during the training, the ALAC system can be incrementally upgraded as the condition changes.As mentioned in the previous chapter, the main challenges of the current anomaly IDS are that the complication of developing a system with these characteristics is higher than in the case of misuse detection (Elhag et al., 2015). Furthermore, a higher percentage of false alarms is raised (Elhag et al., 2015; Shah & Issac, 2018), coupled with a low detection rate (Raghav, 2013; Singh et al., 2015). There is also an issue of the unbalanced dataset which impacts the evaluation of the models (Fossaceca et al., 2015; Tavallaee et al., 2009). This work proposes a new hybrid model comprised of a new machine learning (FLN) and PSO algorithms for the reduction of the false alarm and increasing of the accuracy of IDS using NSL-KDD dataset.

2.3.4 Intrusion Detection System Dataset

In IDS research, the KDD Cup 1999 is the commonest data set used. This data contains about 4,900,000 connection records and each record consists of 41 features ("KDD Cup 1999 Dataset," 2010). This data has been statistically analysed and presented (Tavallaee et al., 2009). In the KDD data set, there are four major categories of attacks as mentioned in previous chapter; they are:

- Denial of Service (DoS):

DoS is a form of attack in which the intruder has access to the computing accessories and make the system too clustered or busy to consider genuine requests, thereby, denying access to the legitimate users.

- Surveillance and Other Probing:

Probing is a situation where an attacker can scan the network and identify system vulnerabilities to exploit based on the gained information.

- Unauthorized Access from a Remote Machine (R2L):

A remote to user (R2L) attack is a situation where a packet is sent by an attacker to a network machine, then, exploit the weakness of that machine to gain an unlawful access to the network as a regular user.

- Unauthorized Access to Local Super User (U2R):

User to root is a situation where an attacker can access a network as a regular user and then, exploit the network susceptibility to getting root access. Many ML and pattern classification algorithms have been used to solve intrusion detection problems based on the KDD dataset, but have all failed to detect most of the remote-to-local and user-to-root attacks. The limitations of the KDD99 data set has been identified (Sabhnani & Serpen, 2004) and suggested not to be used in training pattern recognition or ML algorithms for misuse detection of these two attack categories. NSL-KDD data set, it has been reported that the KDD99 dataset has many problems; for example, it contains several redundant features, and the difficulty level of the different records and the percentage of records in the original KDD dataset are not inversely proportional. These deficits result in a poor evaluation of different proposed ID techniques. The NSL-KDD dataset was proposed to overcome some of these inherent problems of the KDD Cup 1999 data set. The proposed new dataset consists of selected records of the complete KDD dataset (Tavallaee et al., 2009). Table 2.2 showed the NSL-KDD data variables, while Table 2.3 showed the distribution of attack records per attack category (Kang & Kim, 2016). The following are some of the advantages of the NSL-KDD over the original KDD dataset (Tavallaee et al., 2009):

1. Redundant records are excluded in the training set. Thus, there is no bias towards more frequent records.
2. In the original KDD data set, the number of records selected from each group level and the percentage of records is inversely related.
3. If there is a sound number of records in the training and testing portions, experiments on the whole set can be economically tested without the necessity for a random sample at a reduced scale.

Table 2.2 The features of the NSL-KDD data set

F	Name	Type	Min	Max
1.	Duration	Numeric	0	54,451
2.	protocol_type	Symbolic	0	2
3.	Service	Symbolic	0	64
4.	Flag	Symbolic	0	10
5.	src_bytes	Numeric	0	89,581,520
6.	dst_bytes	Numeric	0	7,028,652
7.	Land	Boolean	0	1
8.	wrong_fragment	Numeric	0	3
9.	Urgent	Numeric	0	3
10.	Hot	Numeric	0	101
11.	num_failed_logins	Numeric	0	4
12.	logged_in	Boolean	0	1
13.	num_compromised	Numeric	0	7479
14.	root_shell	Numeric	0	1
15.	su_attempted	Numeric	0	2
16.	num_root	Numeric	0	7468
17.	num_file_creations	Numeric	0	100
18.	num_shells	Numeric	0	2
19.	num_access_files	Numeric	0	9
20.	num_outbound_cmds	Numeric	0	0
21.	is_host_login	Boolean	0	1
22.	is_guest_login	Boolean	0	1
23.	Count	Numeric	0	511
24.	srv_count	Numeric	0	511
25.	serror_rate	Numeric	0	1.0
26.	srv_serror_rate	Numeric	0	1.0
27.	rerror_rate	Numeric	0	1.0
28.	srv_rerror_rate	Numeric	0	1.0
29.	same_srv_rate	Numeric	0	1.0
30.	diff_srv_rate	Numeric	0	1.0
31.	srv_diff_host_rate	Numeric	0	1.0
32.	dst_host_count	Numeric	0	255
33.	dst_host_srv_count	Numeric	0	255
34.	dst_host_same_srv_rate	Numeric	0	1.0
35.	dst_host_diff_srv_rate	Numeric	0	1.0
36.	dst_host_same_src_port_rate	Numeric	0	1.0
37.	dst_host_srv_diff_host_rate	Numeric	0	1.0
38.	dst_host_serror_rate	Numeric	0	1.0
39.	dst_host_srv_serror_rate	Numeric	0	1.0
40.	dst_host_rerror_rate	Numeric	0	1.0
41.	dst_host_srv_rerror_rate	Numeric	0	1.0

Table 2.3 Distribution of Attack Records per NSL-KDD Attack Category

Attack Category	Attack Name	No. of Records
	Back	956
	Land	18
	Neptune	41214
	Pod	201
	Smurf	2646
	Teardrop	892
DoS		**45927**
	Satan	3633
	Ipsweep	3599
	Nmap	1493
	Portsweep	2931
Probe		**995**
	Guess_Password	53
	Ftp_write	8
Table 3.4Continued		
	Imap	11
	Phf	4
	Multihop	7
	Warezmaster	20
	Warezclinet	890
	Spy	2
R2L		**995**
	Buffer_overflow	30
	Loadmodule	9
	Rootkit	10
	Perl	3
Normal		**67343**
U2R		**52**
Total		**125973**

Tavallaee et al. (2009) employed 21 learned machines to label the records of the entire KDD training and testing sets. This labeling provided 21 predicted labels for each record. From the results, they observed that about 98% of the training records and 86% of the testing records were correctly classified with all the learners. Hence, the machines achieve about 86% to 98% classification rate. Note that a minimum classification rate of 86% can cause difficulties in the comparison task since they all vary within 86 to 100%. A significant problem with the KDD99 data set is the large rate of redundancy. An analysis of the KDD99 training and testing sets showed that 78% and 75% of the dataset are duplicated in both sets. This redundancy can cause learning algorithms to be biased towards the more frequent data records and prevent them from learning occasional records such as U2R attacks which are basically more dangerous to the networks. Additionally, this method can lead to biased evaluation results.

Similarly, Sabhnani and Serpen (2004) presented similar problems in a different way. They reported that pattern recognition and ML algorithms trained with the KDD99 training subset and tested on the KDD99 testing data may fail to detect most of the U2R and R2L attacks. Tavallaee et al. (2009) offered a way out of these issues by providing a new set of training and testing sets comprised of selected records of the complete KDD data set. These new data set are free of the aforementioned problems and are referred to as NSL-KDD. It is freely and publicly available for use. The NSL-KDD was streamlined by removing most of the redundant records and adjusting their distribution based on difficulty (Fossaceca et al. 2015). The next subsection provide an explanation of machine learning algorithms and methods based on intrusion detection system.

2.4 Machine Learning Based Intrusion Detection System

The conventional techniques such as encryption, firewalls, and access control have failed in the provision of complete protection to networks and systems from the increasingly complicated forms of attacks and malware (Kaur et al., 2014). Consequently, the IDS have been developed as an indispensable aspect of security systems which is used for the detection of attacks even before they occur (Mishra et al., 2016). There are certain issues to consider when building IDS, issues like data collection, intrusion recognition, data pre-processing, reporting, and response. Among these issues, the most important is intrusion recognition. Audit data are examined and compared with detection models to describe their nature (normal or benign). This is to ensure the identification of both successful and unsuccessful intrusion attempts.

Furthermore, ML has not demonstrated good detection accuracy and fast processing times when challenged with these requirements (Zamani & Movahedi, 2013). Fortunately, (Fossaceca et al., 2015)owing to the ability of the computational intelligence techniques to adapt and exhibit fault tolerance, as well as their high calculative speed and resilience against noisy information, they compensate for the limitations of these two approaches. Also, in section 2.3.2.2 discussed several categorized of algorithms based on IDS and the benefits of ML in compared with other categories.

Although the amount of work that adoption machine learning based on intrusion detection is increasing because of the advantages that mentioned in anomaly based IDS section. Moreover, most of the systems that are based on these techniques are prone to high rates of false positive and false-negative alarm(Perera Miriya Thanthrige, Jagath Samarabandu, 2016). They also lack the ability to continuously adapt to emerging attack behaviors (Udaya et al., 2016). Previously, several ML techniques have been used to solve ID problems with the hope that they will improve the rate and adaptability of detection. It can be summarized that the computational intelligence systems also have similar features such as computational adaptation, high computational speed, fault tolerance, and fewer chances of error due to redundant information. In this work, the IDS-based intelligence systems are classified as singles and hybrids, as discussed in the next subsections.

2.4.1 Single Algorithm Based Intrusion Detection System

Soft computing techniques (SCT) have found application in IDS due to their ability to handle uncertain and partially true data (Moradi and Zulkernine, 2004). Several SCTs exist, such as artificial intelligence (ANN), Association rule mining, Fuzzy logic, support vector machine (SVM), genetic algorithm (GA), etc. which can be deployed for the improvement of the detection accuracy of the signature or anomaly-based IDS. The ANNs are used for ID for data generalization and classification (as normal or intrusive) (Han and Kamber, 2006; Ibrahim, 2010).

Zhang et al. (2010) proposed an IDS based on BP and NN. The proposed system was built with only one hidden layer and consequently, the system was shown to have a good efficiency. Furthermore, they evaluated the system on the complete KDD '99 data set. They also stated that when the number of neurons in the hidden layer is extremely few, the system may have poor network non-linear mapping and fault tolerance, but and if too many, there will be a substantial increase in the learning time. Chen et al. (2005) presented better results (for false positive rates) using SVM compared to ANN. This is because ANN requires a large number of training sets for an effective classification while SVM needs just setting fewer parameters. Meanwhile, the SVM is only ideal for binary data. Nevertheless, the detection accuracy of SVM can be enhanced by combining it with other techniques (Li and Lu, 2010).

Finally, combining approaches to improve the robustness of IDS is not a new idea. In fact, prior research has demonstrated that an ensemble of Neural Networks, SVMs, and Multiple Adaptive Regressive Splines (MARS) performs better than each algorithm individually but is difficult to scale to large datasets (Mukkamala, Sung, & Abraham, 2005). For example, in (Koc et al., 2012) demonstrated the effectiveness of using a Hidden Naïve Bayes approach to Network Intrusion Detection and in his Ph.D. dissertation Koc noted that certain algorithms perform better at detecting specific types of attacks than others (Koc, 2013). Koc suggests that utilization of a combined framework consisting of multiple algorithms that would apply the best performing algorithms for classification of each attack category would be an area worthy of future study. Furthermore, (Fossaceca et al., 2015) this gives a motivation for several works that mention the intrusion detection based on the hybrid system is more accurate and improve the performance than the system based on single algorithm or method as we will explain in the next section.

2.4.2 Hybrid Algorithms Based Intrusion Detection System

Hybrid techniques comprise of a combination of any two or more of the aforementioned techniques (Zamani & Movahedi, 2013). The hybrid intelligent systems serve as alternative methods for the unorthodox handling of the increasingly complex detection problems which borders on data ambiguity, high-dimensionality, and uncertainty. They allow the use of both a pre- and raw data knowledge to generate innovative solutions(Woźniak et al., 2014). Figure 2.5 is a rough depiction of the domains of the hybrid intelligent systems

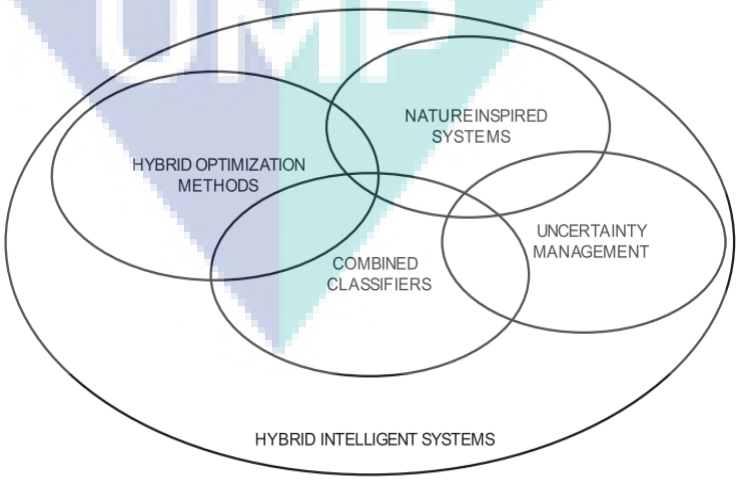


Figure 2.5 Domains of hybrid intelligent systems (Woźniak et al., 2014)

Figure 2.5 showed the domains of the hybrid intelligent systems, and some of these domains deal with data ambiguity and uncertainty using probabilistic or fuzzy representations and feature extraction, while others are concerned with optimization problems encountered in different areas of intelligent system design and problem-solving using either a stochastic process or nature-inspired approach. Finally, classifiers that implement intelligent decision processes are also subject to various forms of hybridization combinations. (Fossaceca et al., 2015) Based on the enumerated benefits of the hybrids systems, several works have proposed their use as IDS.

Hassan (2013) proposed an IDS design based on GA and fuzzy logic using the KDD CUP '99 as a benchmark dataset. The work proposed genetic fuzzy rules for the detection of malicious activities and specific intrusions. In addition, the method was deficient in two aspects, first, it generates false alarms and this is a serious IDS problem; second, it is difficult to generate rules that cover all the attributes of a high dimensional data set. Moreover, Kavitha et al. (2013) also proposed an anomaly IDS based on fuzzy rules. The rules are used to create a detection model in the training phase and used to update the same model during the testing phase. The used Particle Swarm Optimization (PSO) to improve the accuracy of detection by finding the optimum membership functions of fuzzy, they achieved 85% rate of the total accuracy.

In addition, the authors in (Khan, Javed Akhtar, 2016) proposed a popular support vector machine based algorithm called Kernelized support vector machine and an Extreme Learning Machine (ELM) -based algorithm called Kernelized-ELM to work as IDS. They mentioned some limitations of the Artificial Neural Network (ANN) and basic SVM, such as slow learning speed and poor scalability when compared to ELM. A combination of more than one algorithm may be used to eliminate the disadvantages of one another.(G.B. Huang et al., 2012) investigated the use of ELM to classify and detect intrusions. Also mentioned the limitations of SVM and ANN, such as poor performances in multi-class classification, long training times, and requiring parameter training. The results show that basic ELM outperformed SVM in training and testing speed. Based on several previous works, it has been shown that complexity is one of the limitations of hybrid models compared to single models, but

on the other hand, most of the works reported improved accuracy with hybrid models compared to single model algorithms.

2.5 The Artificial Neural Network Versions

A computational network is a style of computation where data flows through a graph and computations happen in the nodes of the graph. The computational networks aim to take feature data and transform same through a network of simple computations to produce one or more outputs. The output is usually decisions based on the input features (Yu et al., 2015). ANNs are computational models used in ML. They are based on a large collection of connected simple units called artificial neurons which have a similar resemblance with the axons in the human brain. The randomization in the neural network (NN) has resulted in three broad families of NN models classified as follows (Scardapane & Wang, 2017):

1. Feedforword Network with Random.
2. Recurrent NN with Random Weights.
3. Randomized Kernel Approximations.

These methods are fundamentally distinct but share two basic ideas that contribute to their efficiency. First, these methods use randomization to define a feature map that transforms the input into a higher dimensional space easy learning. Second, the resulting optimization problem is cast as a standard linear least-squares which, by far, is the simplest scalable and most studied learning procedure to date.

Various methods have been proposed for improving the efficiency of Feedforward neural network (FNN) training. Such methods include subset selection methods (Li, Peng, & Irwin, 2005; Chen, Cowan, & Grant, 1991), second-order optimization methods (Wilamowski & Yu, 2010; Hagan & Menhaj, 1994), and global optimization methods (Yao, 1993; Branke, 1995). Although these methods can lead to faster training speed compared to the BP algorithm, a global optimization solution may still not be achieved with most of these methods. The BP is a first-order gradient method for optimization studies but it is prone to slow convergence and trapping in a local minimum.

Generally, the learning speed of FFNs is slower than normal, and this has been a major setback in their applications over the years. The two reasons for this could be traced to i) the extensive use of the slow gradient-based learning algorithms for training NN, and ii) the iterative tuning of all network parameters using such learning algorithms (Huang et al., 2006). Guang-bin et al. (2004) proposed Extreme Learning Machine (ELM) for the training of Single hidden layer feedforward neural network (SLFNs). In the ELM, there is a random initiation of the hidden nodes before being fixed with no iterative tuning. Only the connection weights between the hidden and output layers are the free parameters that need to be learned. Thus, the ELM is developed as a linear parameter model for solving linear system problems. The ELM, when compared to the conventional NN learning methods, is remarkably more efficient and tends to reach a global optimum. Theoretical studies have demonstrated the ability of ELM to maintain the universal approximation capability of SLFNs even with randomly generated hidden nodes.

(Huang et al., 2006) compared ELM to some common algorithms such as BP and SVM and showed ELM to learn faster than SVM. The analysis also showed ELM to have a better generalization performance compared to BP in most cases. Zhou et al. (2012) investigated the performance of ELM and found it to be independent of the number of hidden neurons if it is reasonably large. Cao et al. (2015) proposed ELM for speeding up the testing process in hybrid Sparse Representation Classification (SRC). The hybrid classifier which is a combination of ELM and SRC showed excellent results in both recognition time and classification rate. The authors justified the use of ELM rather than SVM on the fact that ELM has a faster speed of data processing and generalization performance. Most researches in recent times have depended on the ELM (Kiaee et al., 2015; Wang et al., 2014), and most studies on the comparison of SVM and ELMs have concluded that ELM performed significantly better than SVM in terms of learning/classification and computational speed. Furthermore, next section includes explain of FLN algorithm equations and applications which it is improved most of the limitations that mentioned above.

2.6 Fast Learning Network Algorithm

ANNs have found application in different fields because they can directly approximate complex nonlinear mappings from input patterns (May et al., 2010). The

ANNs have no need for a user-specified problem-solving algorithm, but they can learn from existing examples just like the human brain. Additionally, they have an in-built generalization ability, meaning that they could identify and synchronously respond to similar patterns but not like ones that are used to train ANNs. Meanwhile, the free ANN parameters would be defined by learning from the given training samples based on the gradient descent algorithms which minimizes the learning process (Kiranyaz et al., 2009). Based on these shortcomings, the training of ANNs can take longer time and could have suboptimal solutions.

To address these problems, Huang proposed a new ANN called ELM (Guang-bin Huang et al., 2004) as a novel SLFN where the input weights and bias of the hidden nodes are randomly generated without tuning, and the output weights are analytically determined. The ELM is fast learning algorithm with good a generalization potential and has successfully been deployed in various fields like in function approximation (Approximation & Problems, 2009; Han & Huang, 2006) and pattern classification. The ELM is fully automated and differed from other conventional learning algorithms such as BP which may face difficulties during the manual tuning of its control parameters. However, one problem with ELM is that its classification capability may not be optimal for the learning parameters of the hidden nodes which are randomly assigned but remained unchanged during the training phase (Cao et al., 2012). Therefore, there could be a misclassification of some samples using ELM, especially those near the classification boundary. ELM is also found to require more hidden neurons in many cased compared to other tuning-based algorithms (Huang et al., 2005; Huynh & Won, 2008). However, an increase in the number of hidden layer neurons will cause an exponential increase in the number of weight and random initialization thresholds, though these values cannot be said to be the best parameters that will give optimum performance.

Additionally, the ELM has other deficiencies, such as having more hidden neurons compared to the other tuning-based learning algorithms in several applications (Li, Niu, Duan, et al., 2014). This could make a trained ELM to require a longer time to respond to unknown testing data. Several variants of the ELM have been proposed to address these challenges, such as the incremental ELM (Guang Bin Huang et al., 2006), symmetric ELM (Liu et al., 2013); error-minimized ELM (Feng et al., 2009),

pruning ELM (Rong et al., 2008), two-stage ELM (Lan et al., 2010), voting-based ELM (Cao et al., 2012), ordinal ELM (Deng et al., 2010), evolutionary ELM (Huang et al., 2005), and fully complex ELM (M. Bin Li et al., 2005)

In 2013, the FLN, an artificial neural network-based model was proposed by (Li et al., 2013). The FLN is a double parallel FNN (DPFNN) made up of a parallel connected multilayer feedforward neural network and SLFN (Wang et al., 2011). The output nodes of the DPFNNs not only receive the recodifies external information through the hidden nodes, it also directly receives the external information through the input nodes.

Several studies have demonstrated a better convergence speed and generalization capability of the DPFNN compared to other multilayer FNN (Chen & Chau, 2016). Despite their wide application in several applications, they have certain drawbacks, such as trapping in local minimum and over-fitting. Similarly, many approaches that usually focus on the method for input selection hinder the further minimization of redundancy and improvement of network generalization performance when the DPFNN is composed of MFNN and SFNN with good nonlinear mapping capability and high learning speed. The learning capacity can be improved through a parallel connection between MFNN and SFNN via the indirect and direct information (Rui Huang, 2007). Furthermore, other studies have shown improvement in the speed, accuracy, convergence speed and generalization capability of DPFNN compared to the other common FNNs. The biggest difference with ELM is that the input layer and the output layer are added directly on the basis of the SLFN (G. Li et al., 2017); therefore, the FLN can be considered as a model of nonlinear relation between the hidden layer to the output layer, and the linear relation of the input layer to the output layer. Table 2.4 shows the main differences between ELM and FLN.

Table 2.4 represent FLN vs ELM

FLN	ELM
Double parallel forward neural network. The output nodes not only obtain the recodification of the external information through the hidden layer nodes, but also obtain the external information itself directly through the input layer nodes	Single hidden layer feedforward neural network. The output nodes only obtain the recodification of the external information through the hidden layer nodes

2.6.1 Basic Fast Learning Network

The FLN is made up of a parallel connection of a single layer feedforward neural network and a three-layer feedforward neural network: input layer, hidden layer and output layer(G. Li, Niu, Duan, et al., 2014). The structure of FLN is depicted in Figure 2.7.

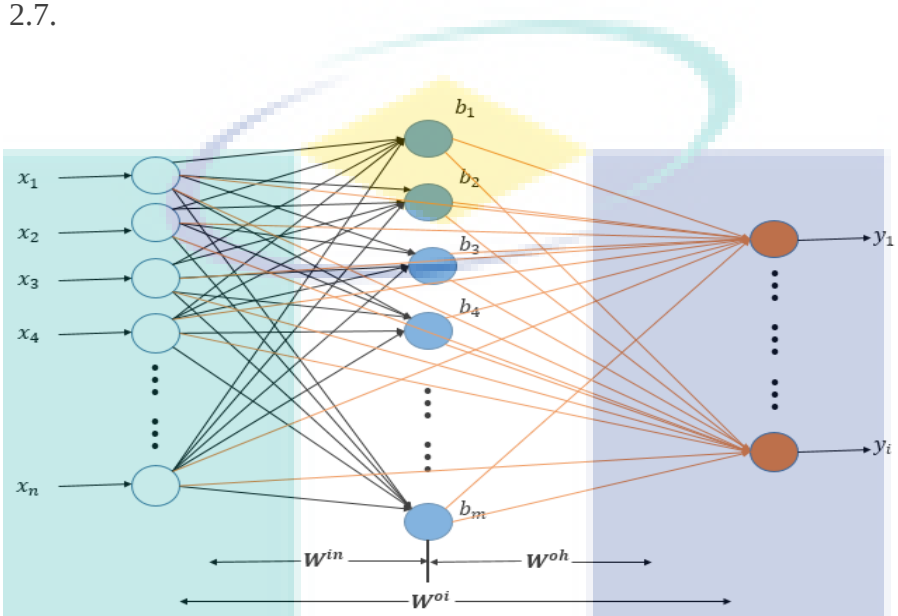


Figure 2.6 The structure of FLN

Suppose there are N arbitrary distinct samples $\{(x_i, y_i),\ i = 1, 2, \ldots, N\}$, in which $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in R^n$ is the n-dimensional eigenvector of the ith sample, and $y_i = [y_{i1}, y_{i2}, \ldots, y_{iI}]^T \in R^I$ is the corresponding I-dimension output vector, the number of hidden layer nodes is m and there are different methods to determining correct number of neurons in hidden layer such as the number of hidden neurons should be between the size of input layer and the size of the output layer (Jeff Heaton, 2008) , $\gamma(.)$ is the active function of hidden nodes, Then FLN is mathematically modeled as:

The equations of deriving the output of the FLN based on the provided matrices and vectors and they are presented in the following equations.

$$y_j = f\left(w^{oi}x_j + \sum_{k=1}^{m} w_k^{oh}\, g\left(w_k^{in}x_j + b_k\right)\right) \qquad 2.5$$

Where j=1,2,….,N, $w^{oi} = [w_1^{oi}, w_2^{oi}, \ldots\ldots, w_I^{oi}]$ is the weight vector connecting the jth output node and input nodes, $w_k^{in} = [w_{k1}^{in}, w_{k2}^{in}, \ldots\ldots, w_{km}^{in}]$ is the weight vector connection the kth hidden node and input nodes, $w_k^{oh} = [w_{1k}^{oh}, w_{2k}^{oh}, \ldots\ldots, w_{Ik}^{oh}]$ is the

weight vector connection the kth hidden node and output nodes, b_k is the kth hidden nodes biases. A more compact representation is given as follows.

$$Y = w^{oi}x + w^{oh}G = [w^{oi}w^{oh}]\begin{bmatrix}X\\G\end{bmatrix} = W\begin{bmatrix}X\\G\end{bmatrix} \quad 2.6$$

Where

$$G\left(W_1^{in},\cdots,W_m^{in},b_1,\cdots,b_m,\cdots,X_N\right) \quad 2.7$$

$$= \begin{bmatrix} g(W_1^{in}X_1 + b_1) & \cdots & g(W_1^{in}X_N + b_1) \\ \vdots & \ddots & \vdots \\ g(W_m^{in}X_1 + b_m) & \cdots & g(W_m^{in}X_N + b_m) \end{bmatrix}_{m\times N}$$

$$W = [W^{oi}W^{oh}]_{I\times(n+m)} \quad 2.8$$

The matrix $W=[W^{oi}W^{oh}]$ is the output weights, and G is the hidden layer output matrix of FLN. A Moore Penrose generalized inverse is used to resolve the model (Liang et al., 2006). The minimum norm least-squares solution of the linear system could be written as:

$$\widehat{w} = (Y)\begin{bmatrix}X\\G\end{bmatrix}^{+} \quad 2.10$$

$$\begin{cases} w^{oi} = \widehat{w}(1{:}I, 1{:}n) \\ w^{oh} = \widehat{w}(1{:}I, n+1{:}n+m) \end{cases} \quad 2.11$$

An algorithm to explain the learning of the FLN is presented in the flowchart depicted in Figure 3.4. The algorithm starts with a random initialization of the weights between the input and hidden layer. Next, the G matrix is found depending on both the input-hidden matrix. This matrix represents the output matrix of the hidden layer. Next, both the input-output matrix w^{oi} and w^{oh} are found using Moore-Penrose equations. As a result, a complete FLN model is established.

The biases of the input weights and hidden layers in the FLN are still randomly generated as in ELM, but the weight values of the connection output-input layer connection and the weight values of the output and input nodes are determined analytically using least squares methods. Therefore, the FLN does not only deal with liner high precision problems but also infinitely approximates nonlinear systems (Li et al., 2017). Additionally, the FLN can overcome the drawbacks of the conventional neural network that does not need iterative calculation. (Li, Niu, & Duan, 2013)

compared FLN with ELM, SVM, and BP, and the results showed FLN with a smaller number of hidden units to achieve a good stability and generalization performance when applied in several applications.

Moreover, there are several works that proposed FLN in different fields, in G. Li et al., 2017) work proposed artificial neural network called Parallel Layer Perceptron Fast Learning Network(PLP-FLN). The authors used Parallel Layer Perceptron (PLP) to map nonlinear input-output relationships. By comparing the PLP-FLN model with FLN, ELM and KELM, through 7 classification problems and 12 regression applications the results showed that the PLP-FLN had demonstrated better approximations, generalization ability and classification performance.

In (P. Niu, Chen, et al., 2017) work, authors proposed model based on adopted an ameliorated krill herd algorithm (A-KH) to adjust the parameters of the FLN in order to obtain a high-precision prediction model. The proposed model used s to build a regression model based the turbine heat rate of a 600MW supercritical steam. The model is compared with several numeric functions to find the best function value based on 8 functions. The total data that used to evaluate the model include 96 pairs for training set, and 24 pairs for testing set, and that mean not enough for evaluate complex propose a model. FLN has been used as a benchmark to compare to the performance of some proposed models (Niu et al. (2016); Li et al. (2017). The FLN is one of the new algorithms based on ANN that was used to build NOx emissions model Ma et al. (2018). Moreover, most of the researchers stated that FLN can overcome the problems of traditional neural network (Li et al., 2017). (Fossaceca et al., 2015) A brief review of other ML methods compared to FLN is provided as follows:

1. Bayes Network Classifier: This classifier depends on probability feature distribution for the estimation of the conditional capabilities of certain observations belonging to certain classes. Even though this method is well suited for large datasets, the classes are presumed to be independent and it is difficult to estimate the actual network traffic probabilities.

2. K Nearest Neighbours (KNN): In this classifier, the majority vote of nearest neighbours is considered during the classification of each observation via a similarity measure. Although this is an easy technique to implement, the KNN requires a

significantly large storage space, performs poorly on datasets with high dimensionality.

3. Decision Tree: The decision tree divides data into specific classes using a recursive approach to learning; however, this algorithm is very complex and unstable.

4. Neural Networks: The NN rely on the modelling of the human brain operation. Its single or multilayer perceptron's are trained with gradient descent algorithms like BP for the optimization of the neuron weights and to minimize the error between the predicted and actual training samples. However, NNs are built with a large computational burden, and they are susceptible to over-fitting and long processing times.

5. Support Vector Machines (SVMs): The SVM uses statistical optimization methods to construct a set of "hyperplanes" in high dimensional space for the classification of network traffic into categories. In the SVM, there is a need to carefully select the kernel type and a proper adjustment of its parameters. The SVMs are highly accurate and can model complex decision boundaries with fewer chances of data over-fitting. However, the SVM is highly complex and requires a large memory space. The selection of the Kernel is usually difficult and the algorithm runs slowly when applied on larger datasets.

6. Extreme Learning Machine: The ELM was proposed to address most of the issues of the common learning methods, such as the number of epochs, learning rate, stopping criterion, and local minima. However, the Elm still has certain drawbacks, such as the need for more hidden neurons in many applications compared to the conventional algorithms. This makes trained ELMs to require a longer time to respond to unknown testing data.

Finally, even FLN algorithm improved most of the above tradition machine learning algorithms. It is still facing several limitations such as random select of the main parameter's values. Moreover, proposed PSO hybrid with FLN to improve the limitation as in the next will include an overview of PSO algorithm and its improvements.

2.7 Particle Swarm Optimization Algorithm

Over the past 2 decades, nature-inspired metaheuristics have attracted much attention due to their efficiency in establishing accurate solutions to complex industrial and engineering problems, especially the NP-complete problems. Most nature inspired metaheuristics are classified as stochastic techniques. These stochastic algorithms randomly pick a set of solutions and improve them based on the algorithmic mechanism. The solutions are constantly improved until a set stopping criterion is met. Stochastic techniques are classified as random searches but guided to the next iteration by heuristics. In the last few years, many stochastic algorithms have been proposed due to their great success in finding best solutions to science and engineering problem (Diao & Shen, 2015; Slowik & Kwasnicka, 2018).

The PSO is one of the most popular algorithms first introduced by (R. Eberhart & Kennedy, 1995). The PSO solves optimization problems by emulating the flocking behavior of birds; where each bird is regarded as a solution. The advantage of the PSO when compared to the evolution-based frameworks like the GA lies in its ease of implementation and in requiring just a few parameters to be adjusted (Satapathy et al., 2014; Shi et al., 2005). The PSO has successfully been applied in several instances such as function optimization, fuzzy systems, artificial neural network training, and feature selection (Chen et al., 2015; Huang & Dun, 2008). It can also be applied to other areas where GA can be employed. The following subsection discussed the original PSO and its variants.

2.7.1 Standard Particle Swarm Optimization (PSO)

The standard version of PSO is a well-known optimization algorithm. The swarm is initialized with a random population of solutions. The PSO searches for the best positions by updating its component generations. The generated particles in the PSO (which are the solutions) fly in a D-dimensional search space at a velocity dynamically adjusted based on both their own respective experiences and the experience of their neighbours. The *i*th particle in the PSO is denoted in the D-dimensional space as $x_i = (x_{i1}, x_{i2}, x_{i3}, \ldots., x_{iD})$ where $x_{id} \in [LB_d, UB_d]$, d $\in [1, D]$, LB_d, UB_d respectively represents the minimum and maximum limits of the dth dimension. The velocity of particle i is given as $v_i = (v_{i1}, v_{i2}, v_{i3}, \ldots\ldots, v_{iD})$, which is

maintained at a maximum user-specified velocity V_{max}. The particles, at each time step *t*, are manipulated based on the following relation:

$$vi\ (t+1) = v_i\ (t) + r_1 c_1 (P_i - x_i\ (t)) + r_2 c_2\ (P_g - x_i\ (t)) \quad 2.12$$

$$xi\ (t+1) = x_i\ (t) + v_i\ (t) \quad 2.13$$

where r_1 and r_2 represents the random values in the range of 0 and 1. c_1 and c_2 represents the acceleration constants that governs the extent a particle can move within a given iteration. The previous best position of the *i*th particle is represented by P_i.

Based on the several definitions of P_g, there are 2 variants of the PSO. A global version of PSO is achieved when P_g represents the position of the best particle among the other particles in the same population (also referred to as the as$gbest$). But if P_g is derived from a few number of adjacent particles of a population (called $lbest$), a local version of PSO is achieved. An inertia term *w* was later introduced by Shi & Eberhart (1998) via a modification of Equation 2.13 into:

$$vi\ (t+1) = w \times v_i\ (t) + r_1 c_1\ (P_i - x_i\ (t)) + r_2 c_2\ (P_g - x_i\ (t)) \quad 2.14$$

They suggested that a proper balance between global and local explorations can be achieved through a proper selection of *w*, thus, requiring averagely less iterations to establish an optimal solution. The *w*, as originally developed, is set using the following equation:

$$w = w_{max} - \frac{w_{max} - w_{min}}{itr_{max}} \times itr \quad 2.15$$

where w_{max} represents the initial weight, w_{min} represents the final weights, itr_{max} is the highest number of allowable iterations, and itr represents the present number of iterations. This version of PSO is henceforth referred as a linearly decrease inertia weight method (LPSO). In addition, in LPSO, a random inertia weight factor for dynamic systems tracking has also been suggested (Eberhart & Yuhui, 2001). This inertia weight factor in this development is set to randomly change based on the following relation:

$$w = 0.5 - \frac{rand()}{2} \quad 2.16$$

where rand () represents a uniformly distributed random number in the range of 0 and 1. The acceleration coefficients were suggested to be maintained at 1.49. This method is henceforth referred to as random weight method (RPSO) in the remaining part of this work.

However, PSO is susceptible to premature convergence, trapping to local optimum, and low convergence speed for multimodal optimization problems (Lu et al., 2017). Several studies have previously focused on the improvement of PSO and have resulted in the emergence of several PSO variants, such as PSO with passive congregation (PSOPC), (He et al., 2004), and the extended PSO (EPSO) algorithms (Jun-jie, 2005). The EPSO introduced a third target point in the formula for the position and velocity of PSO. Moreover, Lu et al. (2017) proposed APSO as an enhancement of the normal PSO by hybridizing the algorithm with the mutation mechanism of GA and standard PSO. The basic concept of the APSO is that the position of the ith particle in the kth generation does not only depend on $xbest, local$ and $xbest, global$ but also on the additional parameter known as an active target point. This proposal improved the velocity update formula of the standard PSO and make it mutative.

Several time-varying strategies have been suggested for the regulation of PSO parameters. These modifications are fundamentally based on the tuning of the learning weights of the particles for their exemplars (Xia et al., 2018). For instance, the ubiquitous parameters update rules introduced by Ratnaweera et al. (2004) and Shi & Eberhart (1998) have three PSO parameters that are adjusted based on the iteration numbers with the aim of meeting different search criteria of different evolutionary stages.

Being that a larger w encourages exploration while a smaller one benefits exploitation, it seems right to deploy a time-varying w to strike a balance between them. The ubiquitous w update rule introduced by Shi & Eberhart (1998) linearly decreases from 0.9 to 0.4 during optimization process and is still applicable in most variants of PSO. Furthermore, Ratnaweera et al. (2004) proposed HPSO-TVAC which is motivated by the iteration based w. Considering the nonlinear and complicated nature of PSO search process, many nonlinear strategies have been introduced for tuning its parameters (R. C. Eberhart, 2001; Pornsing et al., 2016). Although a reliable performance can be

achieved by altering these parameters, their common feature, i.e., iteration-based strategy, does not have a serious significance on their individual evolutionary information. Thus, to maximize the utilization of the historical evolutionary information and provide a better parameters tuning method, several adaptive strategies have been suggested in recent years (Tanweer et al., 2015; Zhan et al., 2009; Limin Zhang et al., 2015). Tanweer et al. (2015) for instance proposed the APSO, in which w, c1, and c2 are dependent on the evolutionary state estimation (ESE). This ESE relies on the population distribution and the fitness of the particles rather than on the number of iterations. With the adaptive strategies, different particles could perform different roles on exploration and exploitation during the search process.

The other variants of PSO proposed in the literature are multi-swarm methods. Niu et al. (2007) proposed an algorithm that is based on a Master-Slave model. In this algorithm, a population is made up of one master swarm and several slave swarms. The particle diversity is maintained by the slave swarms through an independent execution of a single PSO, while the master swarm evolves based on its individual knowledge and that of the slave swarms (Sinan, 2018). This multi-swarm is explained in the next section.

2.7.2 Multi-Swarm PSO Algorithm

The core idea of the multi-swarm is the interaction between several groups while searching for a solution (Okulewicz & Mandziuk, 2015). Many multi-swarm-based schemes have been proposed, each being inspired by a natural behavior. (Sinan Q. Salih1, 2, 2018)proposed a new cooperative multi-swarm scheme inspired by human social behavior (the interaction between a group of people known as 'Clan' and their leaders). The proposed scheme consists of several swarms called clans; each clan consists of several solutions represented by the group members. The best member of each clan is the clan leader and has control over the members of its clan in terms of the time to move and where they are moving to. Figure 2.6 shows the structure of the individual swarms.

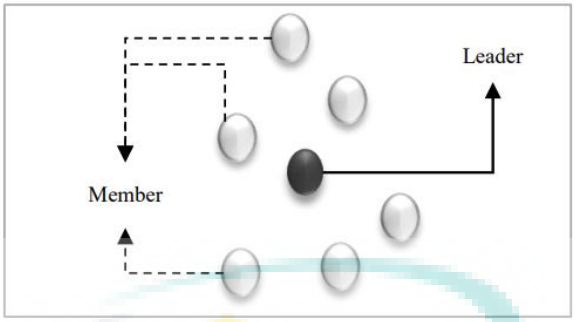


Figure 2.6 The structure of the individual swarm(Sinan Q. Salih1,2 , 2018)

In each generation, the leaders often meet in one room to select an overall best leader who will update the position of the other leaders based on his new-found position. This behavior of knowledge sharing helps to balance the exploration stage with the searching process of the PSO, which represents the exploitation stage. The new multi-swarm approach is called a 'Meeting Room Approach' (MRA). Figure 2.7 shows the MRA model, where each member in the clan represents a particle in the swarm, and its position and velocity updated based on the steps of PSO algorithm. Once the new generation of each clan has been set, a new clan leader (the best leader) is elected and sent to the meeting room. The best among the leaders will be selected as the overall best leader (global best) in the meeting room. The newly-selected overall best leader shares his positional information with the other leaders using the following relation:

$$w^{Ln} = \left(\frac{w^{Lg} - w^{Ln}}{Itr}\right) \times rand() \qquad 2.17$$

$$v_i^{Ln}(t+1) = w^{Ln} \times v_i^{Ln}(t) + rc\left(p_g^L - p_g^L(t)\right) \qquad 2.18$$

$$x_i^{Ln}(t+1) = x_i^{Ln}(t) + v_i^{Ln}(t) \qquad 2.19$$

where Ln represents the normal leaders, Lg represents the overall best leader, x_i^L represents the position of the normal leaders, v_i^{Ln} represents the velocity of the normal leaders, w^{Lg} and w^{Ln} represent the inertia weight of the overall best leader and the normal leaders, respectively.Figure 2.7 shown meeting room approach (Sinan Q. Salih1,2 , 2018).

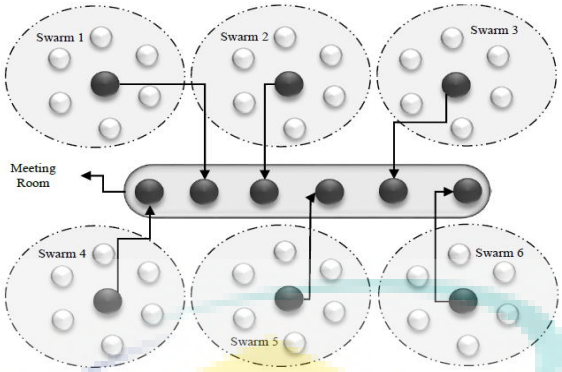


Figure 2.7 Meeting Room Approach(Sinan Q. Salih1,2 , 2018)

After each generation, a new leader is selected for each swarm because the positions of the members are changed or updated during the meeting. The new equation of the inertia in the meeting room controls the exploration of the search algorithm. The pseudo-code for the MPSO algorithm is listed in Figure 2.8.

Algorithm MPSO
Input:
$\#Swarm, \#P, c_1, c_2, \#Dim, MaxGen$
Output:
Best Solution (Leader)
Procedure:
Start
Initialized the swarms
Evaluate the fitness of each particle in swarms
While $(itr \leq MaxGen)$
For each c **in** clans
For each member **in** Clan c
Update the velocity of the member via 2.12
Update the position of the member via 2.13
Next
Select the local best as a $Leader_c$
Next
Select the best Leader among all leaders
Update the inertia weight of the clan 2.17
Update the velocity of the $Leader_c$ via *eq.* 2.18
Update the position of the $Leader_c$ via *eq.* 2.19
Select the Best Leader ever as the global best.
Loop
Return Best Leader
Stop

Figure 2.8 MPSO Pseudo-code

The performance of the proposed (Sinan Q. Salih1,2 , AbdulRahman A. Alsewari1, Bellal Al-Khateeb2, 2018)MPSO was evaluated by benchmarking with two established algorithms, the original PSO (Yuhui Shi & Eberhart, 1999) and the

Master-Slave PSO (MCPSO)(Niu et al., 2007) based on some of the nonlinear benchmark functions (Sphere Unimodal, Griewank Unimodal, Ackley Multimodal). The results It may be concluded that MPSO required less computational complexity, and yet, had a better performance in terms of finding the best solution. Moreover, several works proposed different PSO versions to be hybrid with ANN versions to improve and reduce the randomness impact as shows in the next section.

2.7.3 Artificial Neural Networks Based On Particle Swarm Optimization

The ANNs with 3 layers (hidden, input and output layers) are used to forecast the inputs-outputs relationship in complex and nonlinear engineering systems (Ghaedi et al., 2015). The efficiency of the ANN model depends on certain variables such as the number of output and hidden layers, the nature of transfer function, and the number of nodes in each layer (Raja, 2014). The errors associated with ANNs can be minimized by selecting appropriate adjustable parameters.

The uniqueness, convergence, robustness, existence, and stability of stochastic numerical solvers using ANN models that are integrated both local and global search methodologies have been proven to solve a range of problems based on linear and nonlinear differential equations (Raja et al., 2016). ANNs integrated with PSO algorithm have been used in several case studies (Ahila et al., 2015) and PSO has been confirmed to have the following advantages over other similar optimization techniques:

(i) There are control parameters in the PSO for balancing global and local exploration of the solution space.

(ii) In the PSO, information on previous good solutions is retained and shared by all particles.

(iii) PSO is simple to implement, thereby reduces the computational time and eliminates the need to select the best operator for a given process.

(iv) PSO has a better convergence, accuracy, and speed compared to other nature-inspired optimization algorithms in certain situations.

In addition to the above features of PSO, it is computationally inexpensive because its requirements for memory and CPU speed are low. PSO is also less sensitive to the nature of the objective function, does not need the calculation of derivatives from other particles, and has few parameters to tune (Marinakis and Marinaki, 2010). Moreover, PSO does not require the information of the objective function under testing; it requires only the value, which is used within primitive mathematical operators, hence, leading to low computation (Padhy, 2009). Hence, PSO was proposed for the optimization of ELM parameters to achieve a higher classification accuracy and to estimate the best values for the hidden nodes of the ELM for power disturbances classification as shown in Figure 2.9. The performance of the proposed PSO-ELM was compared to those of BP, probabilistic neural network, and SVM.

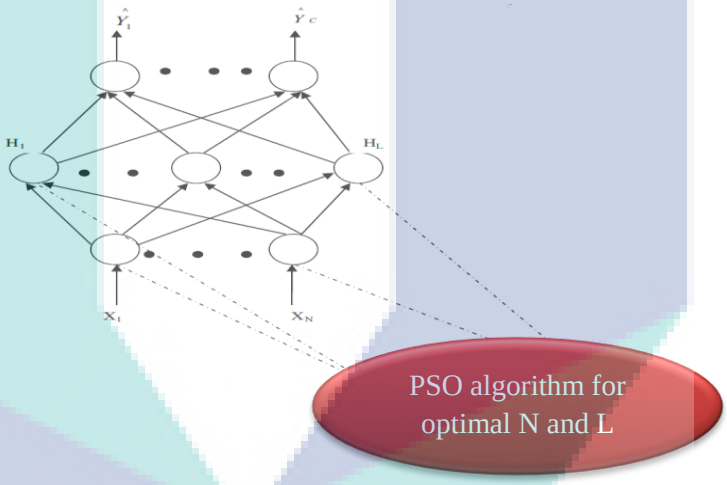


Figure 2.9 Architecture of PSO-ELM classifier

To evaluate the overall performance of the 4 networks, the classifiers were trained with few numbers of training samples and with a large number of the training set. First, 200 datasets were used to train the classifiers, followed by the second phase of testing the classifiers with 500 datasets. The SVM showed a better performance when trained with 200 training datasets compared to BPN, ELM, and PNN. Moreover, as shown in previous sections, the results of this work also proved that the classification accuracy of ELM was better than that of SVM when trained with larger training samples.

Raja (2014) suggested that these solutions can be determined by the integration of the strength of local and global search (PSO) for the optimization of the unknown weights of ANNs. The improvement in the achieved results using these methodologies both in terms of convergence and accuracy, as well as the achieved reduction in computational time and the impact of different designs of the ANN models using appropriate activation functions, are evident. Hence, it is necessary to conduct exploration and exploitation in different ANNs with the appropriate transfer function in the hidden layer in order to develop reliable and effective system models. The learning of the adjustable ANN parameters in this study was done with PSO, active set (AS) and PSO-AS algorithms .The results of the hybrid PSO-AS algorithm were more accurate compared to those of PSO and AS.

In addition, Lu et al. (2017) mentioned that the internal power parameters of kernel-based ELM (KELM) are initialized at random, causing the algorithm to be unstable. Therefore, they proposed the active operator's particle swarm optimization (APSO) to obtain an optimal set KELM. They evaluated the accuracy of APSO-KELM compared to the existing SVM, KNN based on several datasets (Breast, Brain, Colon). Moreover, the authors presented an APSO algorithm that improved on the premature convergence problem and searches performance of the standard PSO.

The APSO is a hybrid combination of the GA's mutation mechanism with the standard PSO. The hybrid APSO-KELM was shown to be relatively stable compared to the component ELM and KELM and achieved a higher accuracy compared to SVM and KNN. The APSO-KELM also had a longer running time. Owing to the introduction of active operators and kernel function into PSO and ELM respectively, the computational complexity of APSO-KELM was increased and this is a major drawback of the hybrid APSO-KELM. Moreover, based on all the works that proposed PSO hybrid with neural network versions gave motivation to used PSO to hybrid with FLN to reduce randomness impact. In the next, section will explain in general the related work of intrusion detection based on hybrid.

2.8 Related Work

There are several proposed of machine learning frameworks that are based on IDS. Xiang et al. (2014) proposed that the current IDS research can be classified into two major domains- anomaly detection and information reduction methods. These

methods mainly focus on the learning methods for alert decision support in anomaly-based ID. The FLN has been earlier demonstrated to perform better than ELM and SVM in terms of training speed, user-friendliness, and accuracy. It has been shown that ML-based ID can use FLN to extend their applicability to significantly larger datasets compared to most of the currently used datasets in most studies. This can be achieved without necessarily increasing the training time due to the near linear scaling ability of the proposed FLN.

Shah & Trivedi (2012) proposed a survey on the ANNs based on IDS and classified the works into simple ANN and hybrid ANN. In the simple approach, they discussed the use of BPNN, SVM, SA, and SOM for anomaly detection. The hybrid approach focused on the use of more than one technique. Jaiganesh (2013) conducted a review of the potential techniques that are based on IDS. The study covered NN, SVMs, and suggested that ELMs are useful for IDS owing to their ease of implementation, fast learning speed, high generalization ability, and working with non-linear kernels and activation functions. Although other studies have suggested the usefulness of ELMs in overcoming most of the discussed challenges (Patel et al., 2012), details of previous studies on ELMs with IDS were not provided. Furthermore, there was no discussion on how to apply ELM on ID problems. They also suggested the chances of overcoming the challenges of the individual algorithms by combining different learning approaches.

Pervez & Farid (2014) proposed an SVM-based filtering algorithm for the selection of multiple ID classification tasks on the NSL-KDD ID dataset. The proposed algorithm achieved 91% classification accuracy when using only 3 input features and 99% using 36 input features, while all the 41 input features of the NSL-KDD set achieved 99% classification accuracy. Meanwhile, the test set performed badly with 0.77. With this level of poor generalization efficiency, this method cannot effectively detect unknown network attacks.

(Fu et al., 2014; Huang et al., 2013) achieved good results with Kernel-based ELM. The kernel selection is a critical step for achieving a good learning performance but the kernel-based ELM usually computes a kernel over the entire input samples and requires much memory (Fossaceca et al., 2015). The computation of large datasets of a full kernel is sometimes not feasible as a result of memory problems, and in the

smaller datasets that executes full kernel computation, there is a need to have a way of combining multiple classifiers or kernels to achieve good results.

In (Fossaceca et al., 2015) explored the feasibility of combining the learning decisions of multi-classifiers for the formulation of a single decision with more accuracy compared to the individual classifiers. This combination of classifiers is motivated by the fact that previous studies have demonstrated a varied classification ability of most classifiers in the detect of specific classes in a multiclass learning problem. The introduction of a novel Multiple Adaptive Reduced Kernel ELM (MARK-ELM)-based IDS made MARK-ELM suitable for the processing of multi-class network intrusion detection systems. Several approaches have shown good detection performance for several attack classes but poor performances for others due to their dependence on KDD '99. The proposed approach achieved a high rate of false positives and a good detection performance which are huge challenges facing network operators.

(Singh et al., 2015) pinpointed large data volumes, low detection rate, and high false alarms as the common challenges of IDS. They used an online based sequential ELM to design an IDS-based anomaly for network traffic analysis. For the performance evaluation of the proposed technique, the standard Kyoto university benchmark dataset was used to test the proposed IDS. The feature that was used in this work was extracted from the KDD data set. The algorithm was not validated on large data sets such as KDD, hence, further validation should be performed.

A heuristic is a way of learning, discovery or problem solving which employs a practical approach that is not guaranteed to be optimal. (Aslahi-Shahri et al., 2015) presented a GA and SVM-based anomaly detection technique. They used GA and SVM for improving the classification performance SVM. The proposed technique was evaluated on the KDDCUP '99 set. As mentioned in the limitations of SVM, it provides a binary classification as normal data or attack. Additionally, the system was only evaluated on the KDD '99 data set.

Furthermore, Table 2.5 shows some of the related works based on IDS. (Vishwakarma, 2017) proposed an Ant Colony Optimization (ACO)-based KNN intrusion detection method. The algorithm was pre-trained with KDD Cup '99 dataset

using ACO, while the performance of the KNN-ACO, BP and SVM were compared based on common performance parameters such as accuracy and false alarm rate. The reported overall accuracy of the proposed algorithm was 94.17%, and its overall FAR was 5.82%. Unfortunately, this algorithm was trained with only 26,167 samples which are relatively a small data volume.

Table 2.5 Relate Works based on IDS

Authors	Model Type	Single	Hybrid	Algorithm	Data set	ACC	Limitations
(Saxena MTech Scholar & Richaariya, 2014)	Anomaly	–	✓	PSO-SVM	10% KDD99	0.993	-the model essay leads to a higher false alarm rate. -The model evaluate based KDD99 with all limitations
(Pervez & Farid, 2014)	Signature	✓	–	SVM	NSL-KDD	0.99	-High rate of false alarm -The performance is worse during the test set - It cannot effectively detect unknown network intrusions.
(Alomari and Othman, 2012)	Anomaly	–	✓	Bees algorithm (BA)+ SVM	KDD cup 99	---	-ELM lower computational requirements than SVMs, -ELMs have shorter training time requirements than SVMs, -ELMs work directly on multi-class classification problems
(Shivhare and Chaturvedi, 2014)	Anomaly	–	✓	BP + DBSCAN algorithm+	KDD cup99	0.9856	-The computational cost using ELM is very small in comparison to back propagation, -Another problem of the conventional back propagation clearing algorithms is slow coverage rate
(Senthilnayaki et al., 2013)	Anomaly	–	✓	GA+ Decision Tree algorithm	KDD cup99	----	it is difficult to precisely model all behaviours since anomaly based detection can detect only known attacks.
(Deshmukh et al., 2015)	Anomaly	–	✓	Naïve Bayes ,Decision Tree	NSL-KDD	0.99	Bayes needs large data sets to work, because the classes are assumed to be independent, and also it is difficult to estimate the actual probabilities of the network traffic.
(Fossaceca et al., 2015)	Anomaly	✓	–	Multiple Kernel-ELM	KDD cup99	0.993	- the author during testing mode didn't depend on the data set testing mode to evaluate the results - This work evaluated based on KDD99, and we mentioned already the problems with this data set.
(Cheng, 2012)	Anomaly	✓	–	ELM	KDD cup99	0.994	-This work used normal ELM with the random select problem. -This work evaluated based on KDD99, and we mentioned already the problems with this data set

Table 2.5 contents shown a summary of some of the related work based on the proposed in this work. The highest accuracy is 0.994 among all works in the table. On another hand, in chapter 4 the experimental results of the proposed model got higher than all of these related works. Moreover, most of these works used the old version of

dataset (KDD99) to evaluate proposed models, and in previous parts of this chapter mentioned some of the limitations that may impact the accuracy credibility.

2.9 Summary

This chapter represents literature review with several sections, begin with general explained for security issues and tools. Next section explained the intrusion detection system types and techniques specially based machine learning and the dataset. Moreover, next sections include analysis of machine learning algorithms such artificial neural networks versions with limitations that lead to proposed fast learning network as core algorithm in this work. In addition, one of FLN limitation is random selection of main parameters value which represents an optimization problem. furthermore, for this reason improved particle swarm optimization proposed to hybrid with FLN based on intrusion detection system.

CHAPTER 3

METHODOLOGY

3.1 Overview

This chapter introduces the developed methodology for this study. It is combined of all steps for developed methodology, dataset preparation and the evaluation measures. Section 3.2 highlights the research phases carried out to reach the target, also explained the main structure of methodology design. Section 3.3 provides dataset pre-processing part and all the detailed steps that have been done for this regard. Next, section 3.4 provides the preface of design methodology. Section 3.5 provides the detailed of proposed FLN. Section 3.6 presents the detailed of optimize fast learning network system based on particle swarm optimization and its applications. Finally, summary for the whole chapter is provided in section 3.7.

3.2 Research Phases

By merge approach the layers one after another the research carried out in few fundamental phases, which are presented in Figures 3.1. Starting from the planning phase the research proceeded to analysis; design and implementation phase and then to the last phase evaluation.

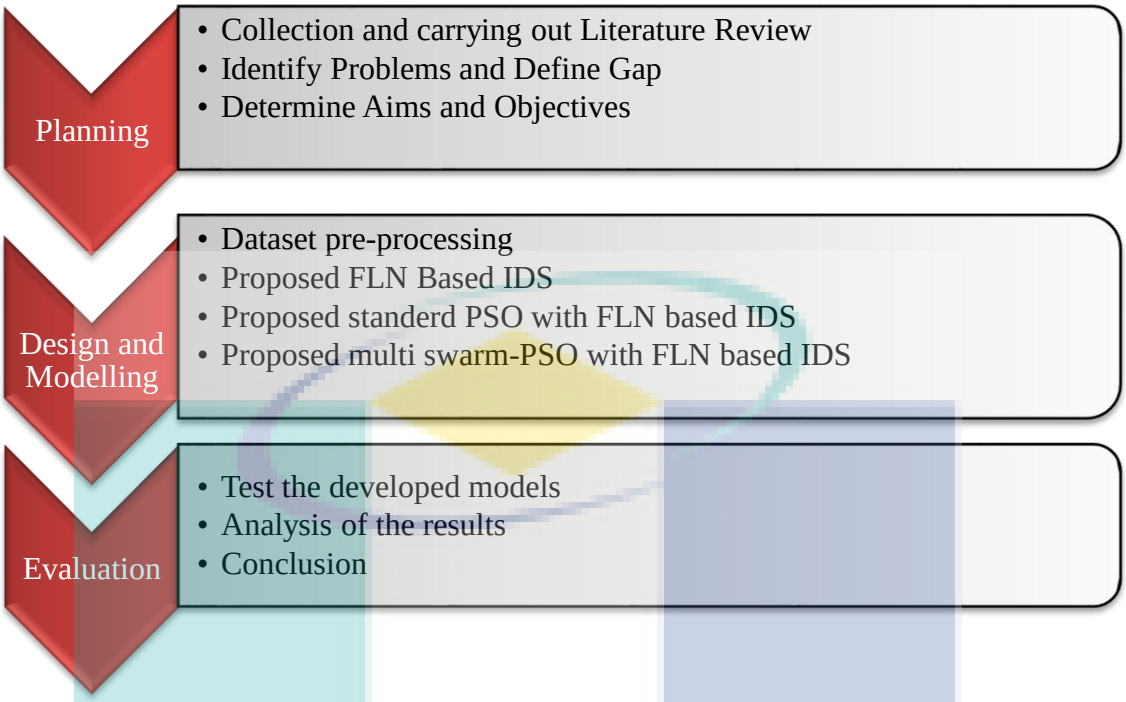


Figure 3.1 Different Phases of the Research

3.2.1 Planning Phase

Planning Phase is the starting phase of the research which consisted of many collection and carrying out the literature review where the main problems are identified first and then precisely identified the research gap. The total articles divided for many groups: one groups of articles discuss the problems in general which is "intrusion detection system based on machine learning ". In another hand, there are many applications based on machine learning algorithms used based IDSs like optimize the main parameters of algorithms, kernel function extension, etc.

At the completion of review stage, it is observed that the previously proposed models of intrusion detection systems are unnecessarily complex to configure. Further, traditional intrusion detection based on machine learning models focuses on computational intensive to reduce the impact of selecting the parameters randomly as its strategy of work in most of the machine learning algorithms, using different techniques and algorithms at different levels. The focus of this research is to reduce the false alarm rate of intrusion detection system, in order to achieve maximum, detect accuracy. FLN algorithm is one of new machine learning algorithm that needs to

adjustment for main parameters(G. Li, Niu, Wang, et al., 2014; P. Niu et al., 2017), this work proposed PSO algorithms for solve this problem.

3.2.2 Design and Modelling

This phase of the research based the problem statement in chapter one, as based on the observation of problem the system is planned to design and then implemented the system through the matlab simulation in order to test the efficiency of proposed system. Firstly, this work tried to reduce the impact of random select of the main parameters of FLN, which its impact the accuracy of the intrusion-detection model.

Secondly, based on the improvement of the PSO, because standard PSO has some shortcomings such as premature convergence and getting stop in local minima. To overcome these shortcomings, many variants of PSO have been proposed (Gülcü & Kodaz, 2015). This work propose a new cooperative multi-swarm scheme inspired by the human social behavior. The proposed scheme consists of several swarms called clans; each clan consists of several solutions represented by the group members. The multi-swarm used as tuning for the main parameters in PSO,(Xia et al., 2018) these parameters adjustments offer relatively reliable performances. Both (PSO, MRPSO) optimization algorithms are also used to adjust the input weights and hidden layer biases of FLN so as to obtain a high-accuracy model of intrusion detection system.

3.2.3 Evaluation Measures

This section reviews the measures, metrics and validation procedures utilized for evaluating the experimental data. Based on the literature research in Chapter 2, most studies report overall accuracy as the main measure of performance for intrusion detection systems. Moreover, some other reviews mentioned other measures, metrics, and validation for evaluating the experimental data. Some studies show more detailed information on the rates of false positive detections and missed detections. These are all useful in evaluating system performance.

The following section formalizes the analysis using standard metrics to objectively evaluate and compare results derived from using different classification methods. Several metrics were utilized to characterize the performance of the system based on the NSL-KDD dataset. A thorough treatment of learning performance

measures can be found in (Singh et al., 2015; Sokolova & Lapalme, 2009), while imbalanced dataset issues are addressed in (Phoungphol et al., 2012). The main metrics employed in this research are:

- **Accuracy***:* is the common metric used for assessing the overall effectiveness of a classifier (Sokolova, M. & Lapalme, G., 2009).

$$\text{Accuracy} = \frac{t_p+t_n}{t_p+f_n+f_p+t_n} \quad 3.1$$

(t_p= true positive, t_n= true negative, f_p = false positive, f_n = false negative)

- **True Positive**: In the context of Intrusion Detection, a "True Positive" is a correct detection of an attack.
- **True Negative**: In the context of Intrusion Detection, a "True Negative" election is the correct identification of "Normal Traffic"
- **False Positive**: Is an indication of an attack on traffic that should have been classified as "normal".
- **False Negative:** In the context of Intrusion Detection, a "False Negative" represents a "missed detection", that is a real attack was misidentified as "Normal" traffic.
- **Standard Deviation:** A quantity expressing by how much the members of a group differ from the mean value for the group, which its calculate as following:

$$\text{SD} = \sqrt{\frac{\sum|x-\mu|^2}{N}} \quad 3.2$$

where ∑ means "sum of", x is a value in the data set, μ is the mean of the data set, and N is the number of data points in the population.

- **Precision**: Refers to the proportion of predicted positive examples that are actually True Positives (Powers, D.M., 2011). Precision can be interpreted as the "Positive Predictive Value" of the classifier(Sun et al., 2007). Precision is calculated as follows (Sokolova, M. & Lapalme, G., 2009):

$$\text{Precision} = \frac{\sum_{i=1}^{C} t_{pi}}{\sum_{i=1}^{C}(t_{pi}+f_{pi})} \quad 3.3$$

where C is the number of classes

- **Recall**: Refers to the proportion of True Positive examples that are detected

(predicted) to be positive (Powers, D.M., 2011). It can be interpreted as the "True Positive Rate" for the classifier (Sun, Y., et. al., 2007). Recall is calculated as follows (Sokolova, M. & Lapalme, G., 2009):

$$\text{Recall} \frac{\sum_{i=1}^{C} t_{pi}}{\sum_{i=1}^{C}(t_{pi}+f_{ni})} \quad 3.4$$

where C is the number of classes

- **F-measure**: Relationship between positively labeled data (e.g. attack) and actual prediction by a classifier based on a per-class average (Sokolova, M. & Lapalme, G., 2009):

$$\text{F-Measu} = \frac{2\times precision\times Recall}{precision+Recall} \quad 3.5$$

- **Detection Rate** It is computed as the ratio between the number of correctly detected attacks and the total number of attacks(Sivatha et al., 2012).

$$\text{Detection Rate(DR)} = \frac{TP}{TP+FP} \quad 3.6$$

- **False Alarm Rate**: It is defined as the ratio between the number of normal instances detected as attack and the total number of normal instances(Sivatha et al., 2012).

$$\text{False AlarmRate(FAR)} = \frac{FP}{FP+TN} \quad 3.7$$

In order to validate the results, (Rodríguez et al., 2010; Singh et al., 2015) k–fold cross validation technique is used for performance evaluation. In this technique, dataset is randomly divided into k different parts. For each iteration, one part is selected as testing and all other (k-1) parts are treated as training dataset. All the connection records are eventually used for training and testing. For all experiments, value of k is taken as 5 because of low bias, low variance, low overfitting and good error estimate (Rodríguez, Pérez, & Lozano, 2010).

3.3 Dataset Preprocessing and Partitioning

The whole dataset is pre-processed in this stage. It consists of two steps, scaling and normalization. In the scaling step, the dataset is converted from string representation into numerical representation. For example, the class label in the dataset contains two different categories 'Normal' and 'Attack', after implementing this step this label is changed into '1' and '0'. Where '1' means normal case, while '0' means attack. The second step is normalization. (Jahan et al., 2011) The normalization cleans the

noises from the dataset, and decreases the differences in the ranges between the features. In this work, we have used Max-Max normalization method, as follows:

$$F_i = \frac{(F_i - Min_i)}{(Max_i - Min_i)} \qquad 3.8$$

Where F_i represents the current feature needs to be normalized, Min_i and Max_i represent the minimum and the maximum value for that feature respectively. The objective function represents the accuracy of the neural network when it is evaluated on the validation set. The validation set is part of the training set. In order to make the validation fairer, *K*-fold validation can be used. The value *K* is determined according to the size of the dataset. Through the cross-validation processes the data is tested to ensure the compatibility of the testing and training subsets and to reduce data discrepancies and its effects on the FLN design. Thus, in cross-validation the first 20% of the data in each run is used as a testing set while the 80% is used for training. After that, a second 20% is used for testing while the remaining 80% is used for training. This process is continuously repeated, taking all possibilities to ensure reliability of the predictive results. Figure 3.2 shows the data partitioning process.

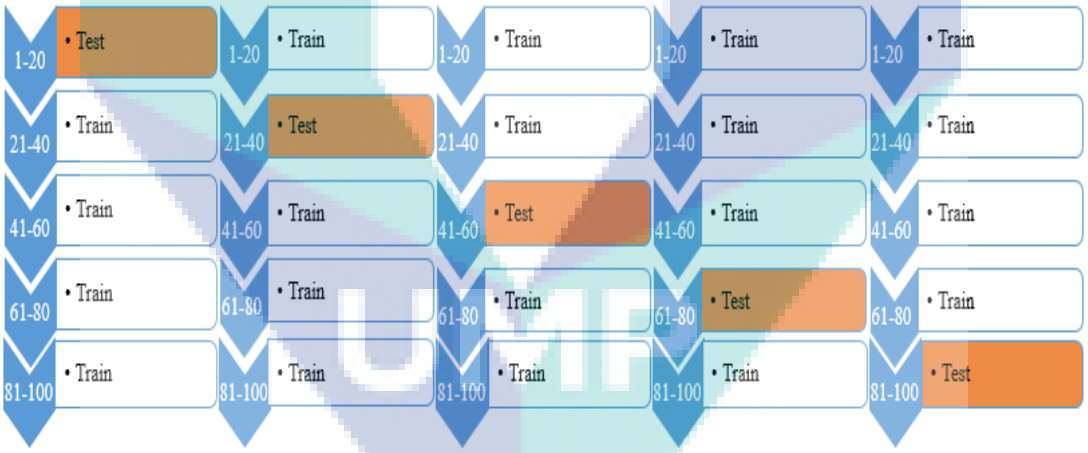


Figure 3.2 Cross validation data partition process

The NSL-KDD data set was used to evaluate the performance of proposed models. The NSL_KDD data set consists of several types of features such as symbolic, numeric and Boolean with varying resolution and ranges. For this study we used all 41 attributes of the data. As of NSL-KDD dataset consist of 148517 number of connection records in both training and testing set with same number of attributes.

3.4 Preface of Design Methodology

This section provides the mathematical methodology for achieving the objectives of the thesis. It is combined of three sub-sections; each one is dedicated for one objective from the objectives of the study. A block diagram of the general developed methodology is depicted in Figure 3.3. It is combined of all sub-blocks of the developed systems with their evaluation part. Also, it shows the mapping of those sub-blocks to the objectives of the thesis. A detailed description of each part is depicted in the following sub- parts in following of the chapter.

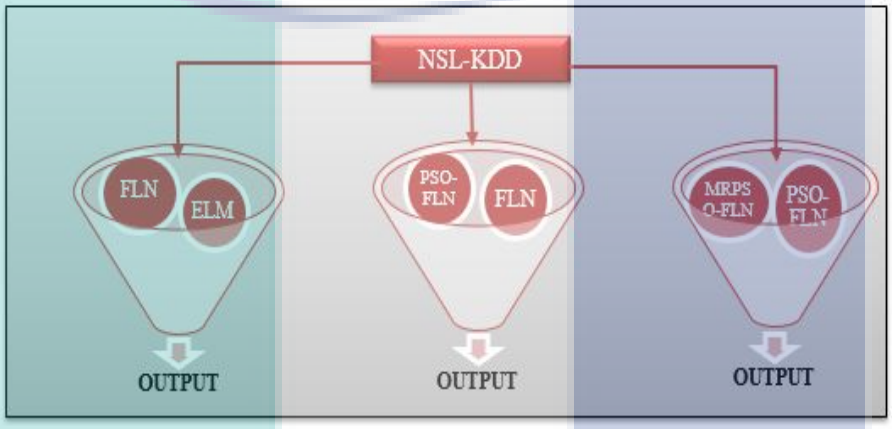


Figure 3.3 The main structure of Methodology

This chapter presents the methodology that developed for implementing the objectives in the thesis. Besides, the evaluation methods and measures are provided. As it has been presented in previous section 3.2.3, this research presents developed of machine learning for detection and identification of intrusions in networks. The main methods are based on fast learning network. More focus is made on optimization method based variants due to the efficiency of fast learning network in performing separation between samples from different classes in compared with ELM. Moreover, this work proposes to adjust the parameters of the FLN and based on PSO and multi swarm-PSO. The evaluation of the developed models is based on benchmark IDS dataset (NSL-KDD).

The chapter starts with general description about the preparation of the IDS dataset. Next, the methodology of implementing the first objective, which is developing a fast learning network based on IDS. Moreover, the second part which

provide the PSO optimization algorithm to adjust the input weights and hidden layer biases of FLN to overcome the impact the random selection of FLN parameters. Third part which president the enhancement of standard PSO by using multi swarm for tuning the parameters of PSO called (MRPSO), and also used to adjust the FLN parameters and evaluate MRPSO-FLN by compare with PSO-FLN based on NSL-KDD dataset. Finally, an evaluation process is performed for the developed model. The evaluation starts with generating the standard measures of evaluating a machine learning model in general and an IDS system in particular.

3.5 Proposed Fast Learning Network

After processing the dataset, this step is to determine the network structure. Typically, in this step, the number of inputs, hidden layers and outputs are selected, as well as the number of neurons selected for each layer based on the intended application. Also, in this study, the model structure consists of a single layer for input, output and hidden as showed in table 2.2. In addition, the input layer contains of 41 neurons based on the features of dataset. And the output layer contains five neurons based on the attacks classes to representing the network output. Furthermore, the study investigated a different number of hidden neurons to verify its effect on the proposed learning algorithms and FLN performance. On the other hand, the choice of activation function plays a role in the convergence of learning algorithms and FLN performance. Finally, the study chose the sigmoid function as an activation function. The configuration of FLN indicates to the needed information to create both the topology of the network and the mathematical structure (activation function formulas) as in table 3.1.

Table 3-1 configuration of FLN

Input	Hidden Layer	Output	Data Indices
$\boldsymbol{x_i=[x_{i1},x_{i2},\cdots,x_{in}]^T \in R^n}$	1,…., m neurons	$y_i=[y_{i1},y_{i2},\cdots,y_{il}]^T \in R^l$	1,2, … N

The learning for FLN could the summarized as Follows:

Step1: Randomly generate the input weights matrix $\boldsymbol{w^{in}}$ and bias matrix **b** based on following equations.

$$w^{in} = (U_{w^{in}} - L_{w^{in}}) \times Rand + L_{w^{in}} \quad 3.9$$

$$b = (U_b - L_b) \times Rand + L_b \quad 3.10$$

Where w^{in} is the weight connecting the hidden node and input nodes, $U_{w^{in}}$ represents the upper bound; $L_{w^{in}}$ represent the lower bound, and $Rand$ represents random value between [-1.5,1.5] based on most of related works. b is the biases of hidden layer nodes; U_b represents the upper bound; L_b represent the lower bound.

Step2: Calculate the hidden output matrix **G** using Eq.2.7

$$G(W_1^{in},\cdots,W_m^{in},b_1,\cdots,b_m,\cdots,X_N)$$

$$= \begin{bmatrix} g(W_1^{in}X_1 + b_1) & \cdots & g(W_1^{in}X_N + b_1) \\ \vdots & \ddots & \vdots \\ g(W_m^{in}X_1 + b_m) & \cdots & g(W_m^{in}X_N + b_m) \end{bmatrix}_{m\times N}$$

Where each position of above matrix, represent the collect of the w^{in} input weight and (b)basic of the hidden layer with (x_i) samples that presented from the dataset. **G** is called the hidden layer output matrix of FLN.

Step3: Calculate the combination matrix **W** using Eq.2.10

According to the Moore-Penronse generalized inverse, the minimum norm leas-squares solution of linear system could be written as:

$$\widehat{w} = (Y)\begin{bmatrix} X \\ G \end{bmatrix}^+$$

Step4: Determine FLN's model parameters based on Eq.2.12

$$\begin{cases} w^{oi} = \hat{w}(1{:}I, 1{:}n) \\ w^{oh} = \hat{w}(1{:}I, n+1{:}n+m) \end{cases}$$

As seen from the above learning process, as the FLN is a parallel connection of a single layer feedforward neural network and a multilayer feedforward network, the output layer nodes not only get the recodification of the external information through the hidden layer nodes, but also get the external information itself directly through the input layer nodes. As the following figure 3.4, which represent the flow chart of FLN algorithm.

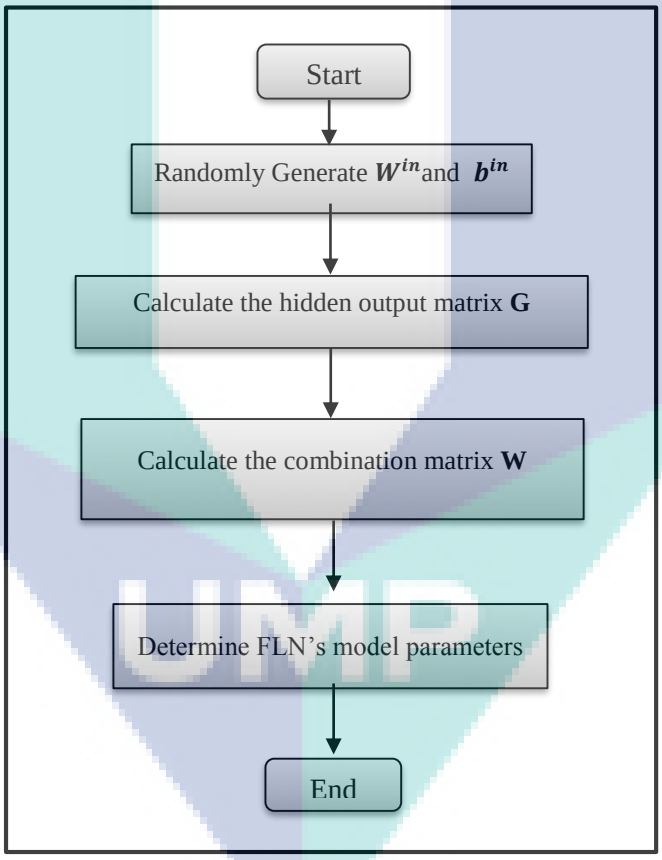


Figure 3.4 the flowchart of the learning model of the FLN

In addition, many literatures have shown that a single layer feedforward neural network in solving the linear problem with higher efficiency, a multilayer feedforward network can very well realize the complex non-linear mapping from the inputs to the outputs. Then, the FLN has the advantages of the two neural networks, but the ELM

does not. So, the FLN with a same or a smaller number of hidden units can achieve much better generalization performance and stability than ELM. In addition, in FLN, the input weights and hidden layer biases are randomly assigned, and the other weights could be analytically determined by least squares methods.

3.6 The Model of FLN Training Based PSO

It may be right to question the rationale for introducing another algorithm to train FLN since there are several algorithms which have been used already as mentioned in chapter 2. Moreover, the PSO algorithm for training FLN is proposed in an effort to complement evaluate for the existing algorithm.

3.6.1 The Definition of the Solution Space

The solution space represents the weights between the input and hidden layers, and the biases of the hidden layer neurons. In the classical fast learning network, such solutions are initialized in a random manner which might cause deviations from the optimal solutions. In the optimized FLN, two matrices are selected as the solution space. As following the structure of FLN weights in table 3.2.

Table 3.2 configuration of FLN

WHI		WOH	WOI
Input Features(n)*Hidden Neurons 41*m	Basis M	Hidden Neurons*output m*5	Input Features(n)*output 41*5

where, m denotes the number of hidden neurons, n denotes the number of inputs, W denotes the input-hidden matrix . The solution space will be as combination of the two matrices.

Start with inputs of the model before explain the steps of proposed, pre-processing for NSL-KDD dataset. The inputs for PSO algorithm include, PSO particles, (c_1, c_2) which represents acceleration factors known as cognitive and social parameters and w is the inertia weight parameter. As for FLN should be represents number of hidden neurons as this work the structure of FLN represent with different number of hidden neurons (10-25-35-50) to analysis the effects of different FLN structure in each model. FLN based PSO

3.6.2 FLN Based PSO

As mention in the previous chapters in FLN section, that it is consists of three layers (input, hidden and output). These layers are contacting by using weights, and biases. In standard FLN, both weights and biases are generated randomly, which may effect on the performance of the classification process. Therefore, generating the best values for them is an issue. In this section, the particle swarm optimization (PSO) is used for finding better values, for both weights and biases. The proposed algorithm called PSO-FLN, which consists of five stages figure 3.6 show the block diagram of PSO-FLN. The learning for PSO-FLN could the summarized as Follows:

Step1: Input

This stages are divided into three parts, PSO parameters, FLN parameters, and dataset. In the first, the main parameters of PSO algorithm are defined, they are cognitive parameter (c_1) , social parameter (c_2) ,inertia weight (w) and number of swarm size (S.S). In the second part, the number of neurons in the hidden layer (m) is defined. The third part the dataset, it represents the dataset used in this thesis, which is NSL-KDD dataset.

Step2: Initialization

Each particle in PSO represent a solution, which consists of two parts, weights and biases the total number of variable is equal to:

$$\text{No.Vers} = \times 2 \qquad 3.12$$

Where m is represents the number of neurons in the hidden layer. When the number (2) represents the main parameters of basic(FLN) (W_m^{in}, b_m) equal to neurons. The solution representation is given in figure 3.5 as following:

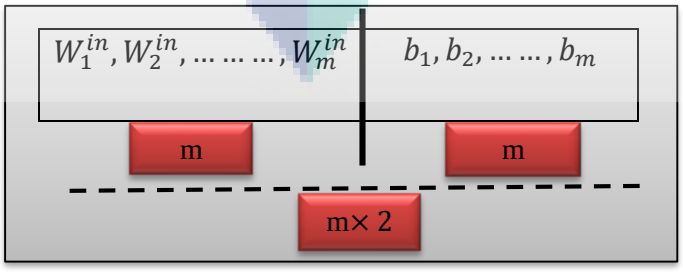


Figure 3.5 Solution Representation (PSO-FLN)

Each variable (position) in particle is initialized by using as following:

Firstpart: $X_i^w = (U_w - L_w) \times Rand + L_w$ 3.13

Secondpart: $X_i^b = (U_b - L_b) \times Rand + L_b$ 3.14

Where X_i^w represents input weight , X_i^b input basis. U_w , U_b in the equations represents the upper boundaries, L_b , L_w represents the lower boundaries. Rand represents a uniformly distributed random number in the range of 0 and 1.

Step3: Fitness Function

In this stage all particles are evaluate by using the fitness equation:

$f(x) = 1 - A$

3.15

Where $f(x)$ represent the error rate of the classification process, thus, finding lower error rate is the main aim of PSO-FLN. Therefore, this is a minimizing problem. And (A) represent the correctly classification (accuracy) sample by using FLN which is given in 3.16.

$$A = \frac{The\ correct\ classification}{N} \quad 3.16$$

Step4: Position Update

In this step, each particle update its position The new particle positions can be calculated using Equation (2.14) and (2.15). After updating the position on the following relations:

$vi\ (t+1) = w \times v_i\ (t) + r_1 c_1\ (P_i - x_i\ (t)) + r_2 c_2\ (P_g - x_i\ (t))$

$x_{iw}\ (t+1) = x_{iw}\ (t) + vi(t)$

After updating the positions of the particle, calculate the fitness value based on a new position and compare the current best with the global best (step t).

Step5: Check Boundaries

The positions of each particle should be checked for any exceeded in the upper or lower boundaries. Therefore, they should stay inside the search space of boundaries

$$x_i = \begin{cases} U_b & , x_i > U_b \\ L_b & , x_i < L_b \end{cases} \quad 3.18$$

Where U_b is represent the upper boundaries; L_b is represent the lower boundaries

Step6: Termination Condition

For each iteration, the global best solution (gbest) is determined. If number of iterations is reach to maximum number of iterations, then stop the searching process and return gbest.

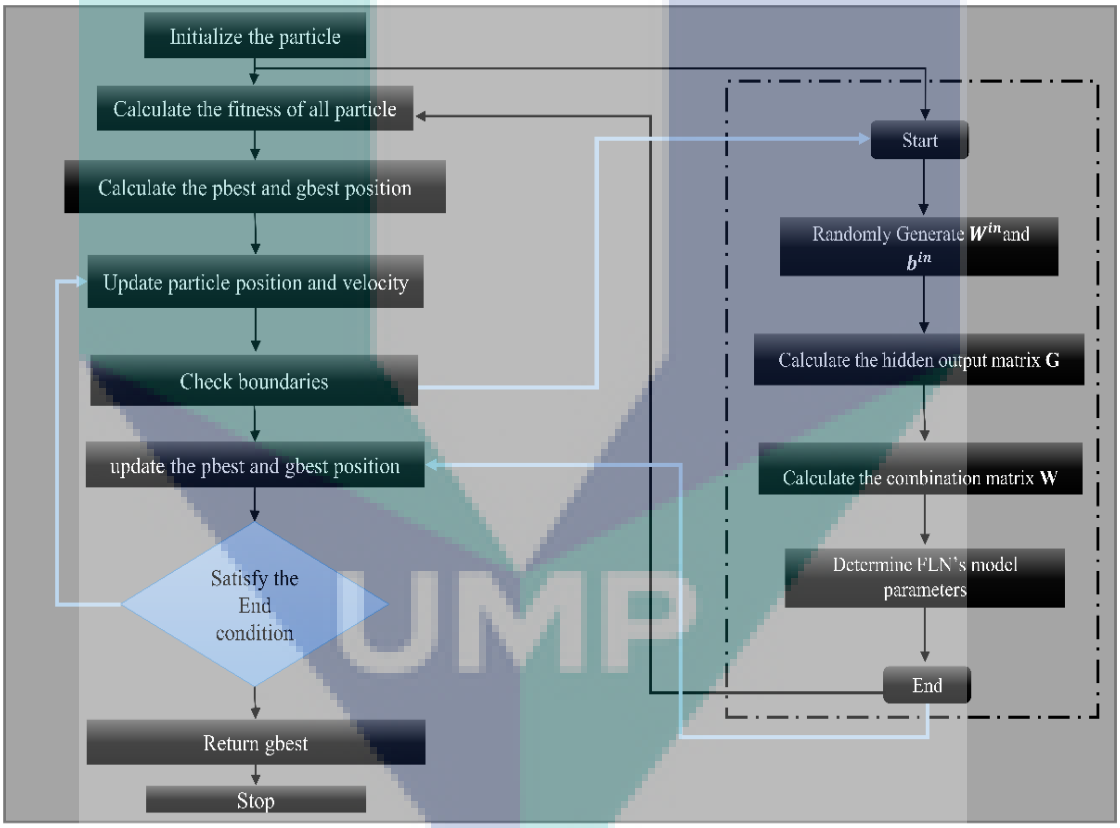


Figure 3.6 PSO-FLN block diagram

3.6.3 FLN based Multi-Swarm Optimization(MRPSO-FLN)

Despite the success of the previously proposed (PSO) algorithm to training FLN, it's still prone to several inadequacies such as, the most ubiquitous update rules

of PSO parameters introduced in, three parameters involved in PSO are adjusted based on different ways aiming to meet different search requirements of different evolutionary stages, the fundament thought of these modification is tuning particles learning weights for their exemplars (Xia et al., 2018). These inadequacies can be solved or enhanced using a new algorithm for the training FLN called multi-swarm optimization (MRPSO). The new proposed MRPSO-FLN shown as following

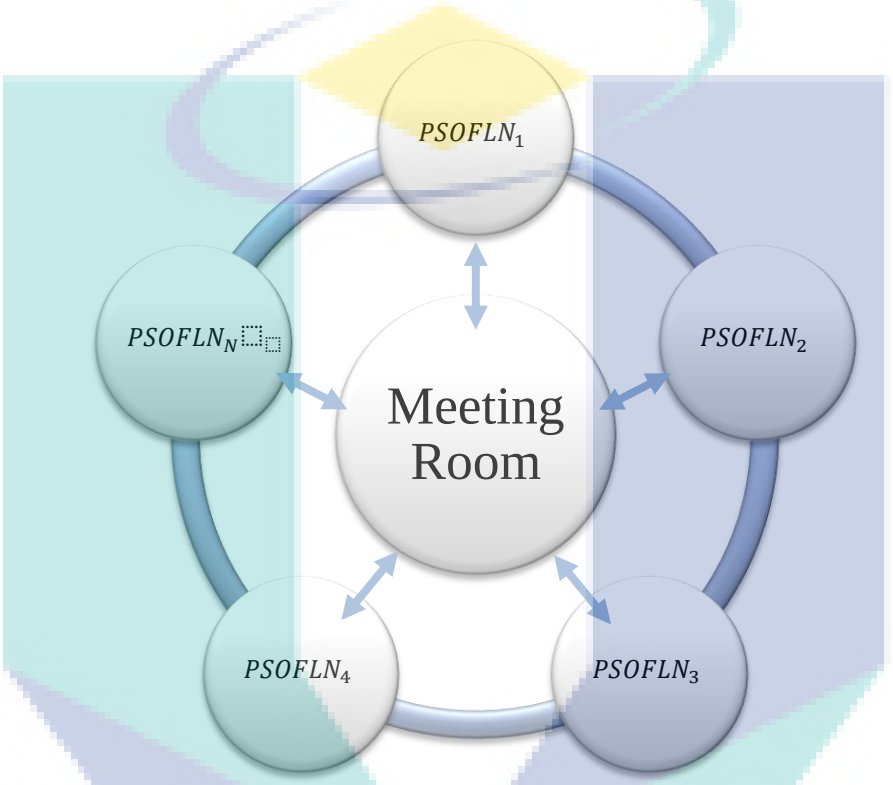


Figure 3.7 Meeting Room Approach

In Figure 3.7, represents the new approach (Meeting Room) approach based standard PSO. As mentioned in previous chapter, the original approach idea focused on update the positions as the group's leaders based on the new positions of the best leader in order to control the balance between exploration and exploitation in the standard PSO.

In this work, in each generation the leaders often meet to select an overall best leader who will update the parameter's values of the other leaders based on his new-found parameter's values, and the comparisons between the leaders depend on fitness function (Accuracy). Moreover, most of the process steps of this model based on previous (PSO-FLN) with some different, will be summaries as following.

Step 1: Initialization

This step divided into two parts, first part, which contains the initialization for the weight, and basis represents FLN parameters same like a previous step. In addition, this step includes initialization for PSO control parameters as following.

$$c_1 = (U_{c1} - L_{c1}) \times Rand + L_{c1} \quad 3.19$$

$$c_2 = (U_{c2} - L_{c2}) \times Rand + L_{c3} \quad 3.20$$

$$w = (U_w - L_w) \times Rand + L_w \quad 3.21$$

c_1 and c_2 represents the acceleration coefficients that governs the extent a particle can move within a given iteration. w represents an inertia weight U in the first equations the upper boundaries, L represents the lower boundaries. Rand represents a uniformly distributed random number in the range of 0 and 1. The solution representation is given in figure 3.8 as following:

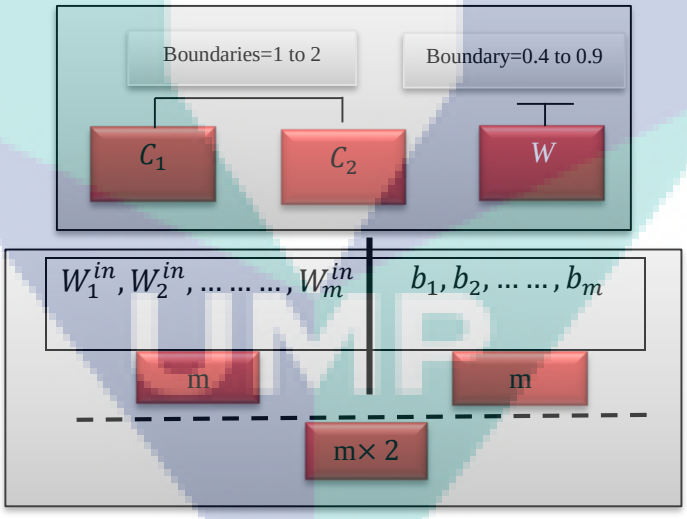


Figure 3.8 solution representation (MRPSO-FLN)

Step 2: Fitness

This step is his same equation 3.15 and strategy as in PSO-FLN process.

Step3: Parameters update

This section represents an update of parameters values based meeting room approach as shown in figure 3.7. Where each member in the clan represents a particle in the swarm and in each generation, each clan will produce leader based fitness function comparison in each iteration and send to meeting room, the leaders after meet to select overall best leaders who also will update parameter's value of other leaders based on a new-found parameters value. The best among the leaders will be selected as the overall best leader (global best) in the meeting room. The newly-selected overall best leader shares his parameter's information with the other leaders using the following relation:

$$C_1^{Ln} = \left(\frac{C_1^{Lb} - C_1^{Ln}}{2}\right) + C_1^{Ln} \quad 3.22$$

$$C_2^{Ln} = \left(\frac{C_2^{Lb} - C_2^{Ln}}{2}\right) + C_2^{Ln} \quad 3.23$$

$$w^{Ln} = \left(\frac{W^{Lb} - W^{Ln}}{2}\right) + W^{Ln} \quad 3.24$$

Where $C_1^{Ln}, C_2^{Ln}, w^{Ln}$ represents the parameters of normal leader in the meeting room, $C_1^{Lb}, C_2^{Lb}, w^{Lb}$ is represents the best leader into the meeting room. The results from these equations, gives the best parameters values which impact on the acceleration velocity of PSO to reach the solution with fewer numbers of iterations. Moreover, in following example shown meeting room approach based above equations:

Accuracy	P_1 **=0.71**	P_2 **=0.81**	P_3 **=0.75**	P_4 **=0.77**
Particles Parameters	C_1=1.0201	C_1=1.5321	C_1=1.9021	C_1=1.6645
	C_2=1.3118	C_2=1.5181	C_2=1.8811	C_2=1.7122
	W_1=0.681	W_1=0.712	W_1=0.886	W_1=0.781

Based on accuracy values of each particle, can observe P2 represents the best leader with best accuracy among other leaders, in following recalling for equations 3.22, 3.23 and 3.24, for comparison between the best leader and normal leaders.

$P_{1.c1} = \left(\frac{1.5321-1.0201}{2}\right)$ +1.0201=1.2761

$P_{3.c1} = \left(\frac{1.5321-1.9021}{2}\right)$ +1.9021=1.7171

$P_{4.c1} = \left(\frac{1.5321-1.6645}{2}\right)$+1.6645=1.5983

$P_{1.c2} = \left(\frac{1.5181-1.3118}{2}\right)$ +1.3118=1.4149

$P_{3.c2} = \left(\frac{1.5181-1.8811}{2}\right)$+1.8811=1.6969

$P_{4.c2} = \left(\frac{1.5181-1.7122}{2}\right)$+1.7122=1.6151

$P_{1.w1} = \left(\frac{0.712-0.681}{2}\right)$+0.681= 0.696

$P_{1.w3} = \left(\frac{0.712-0.886}{2}\right)$+0.886=0.799

$P_{1.w4} = \left(\frac{0.712-0.781}{2}\right)$+0.781=0.7465

After finished the calculation of these equations, each clan start works again with the new values. Moreover, these calculations are repeating based on the iteration's number. The results of equations are shown the new parameters values of normal particles nearer to parameter's values of a best particle. The other steps are steady which it same process in PSO-FLN model

3.7 Summary

For intrusion detection, there is no explicit approach of guiding the IDS system to recognize the lunched attack from the nature of the network traffic because there is huge number of features with non-clear relationship between their values and the corresponding attack that they have resulted from. Therefore, developing a machine learning based IDS system is useful for replacing classical IDS. However, the huge size of the dataset and the non-direct discrimination aspect of it has created a big motivation to develop machine learning based IDS with high performance standards.

Therefore, this chapter has followed an insight procedure to replace previous models with more capable type of machine learning models using three concepts:

firstly, replacing the topological structure of the neural network with more connected type of structure called fast learning neural network. Secondly, optimizing the parameters of the adopted fast learning network to assure improved performance based standard particle swarm optimization, which contribute in the IDS performance as a whole. Thirdly, developing new. a new cooperative multi-swarm scheme (Meeting Room Approach' (MRA) with standard PSO proposed to optimizing fast learning parameters based intrusion detection system. More concentration on our adopted dataset NSL-KDD has been made. Finally, all needed evaluation measures for validating and comparing the system with other systems were introduced with their detailed mathematical equations. In the next chapter, the results of the developed system are presented with thorough discussion and analysis.

CHAPTER 4

RESULTS

4.1 Overview

This chapter presents the results of the methodology that has been presented in the previous chapter. Section 4.2 provides the introduction, which its content the main structure of this chapter. Next, section 4.3 provides the results of the basic FLN with a comparison with ELM and shows optimized FLN results. The different evaluation measures that have been presented in the previous chapter are presented with the confusion matrices. Next, section 4.4 provides the comparative of proposed models. Section 4.5 validates the results for the proposed models are provided. Finally, a thorough discussion and summary for the whole chapter is provided in section 4.6.

4.2 Preparation for Results Structure

This chapter presents the evaluation results and analysis for each model proposed in this work based on the methods that has been declared in previous chapter. In this chapter, the evaluations of the results are represented into three parts. Firstly, this part represents the proposed of basic FLN algorithm based on the intrusion-detection system, which it's compared with basic ELM. The models represented with a different number of neurons in the hidden layer to measure the effect the diffraction of the number of the neurons on the model's accuracy.

Secondly, represented the effect of reduce the impact of the selection randomly of the main parameters of basic FLN, which may affect to the performance of the model. Propose the Particle swarm optimization algorithm to provide the parameters for FLN and trained it. The new mode called PSO-FLN, which evaluate with different numbers of iterations, different number of neurons in hidden layer, and the different

number of particle swarms, to measures the effects of these differentials based on the model performance. The basic PSO parameters were used as selected in (Sivatha et al., 2012) as mentioned in section 4.3.2.

Thirdly, the results of improved particle swarm optimization that provide by meeting room approach to tuning the standard algorithm parameters. The new model called Multi Swarm Optimization (MRPSO) which includes the integrate between the meeting room approach and standard particle swarm optimization. The new model called MRPSO-FLN, also evaluate with different numbers of iterations, different number of neurons in hidden layer, and the different number of particle swarms, to measures the effects of these differentiations based on the model performance. Moreover, for the proposed models, sigmoid represents as activation function for all proposed models. The number of neurons in hidden layer are represented into a different number such as 10-25-35-50 to measure the effect of the number of neurons on the accuracy of models. Moreover, for the optimization models are utilized different number of the iterations 100,250,500, and different number of particle swarms size to analysis and investigate the effect of these differentials on the accuracy of models.

4.3 Results of FLN Models

4.3.1 Results of ELM Vs FLN Comparison

In order to validate the efficiency of FLN based classification NSL-KDD data set, results of accuracy as the best and mean of all runs, detection rate (DR), false alarm rate (FAR), recall, precision, F-measure (F.M), Maximum accuracy (MAX.Acc), Average accuracy (AVR. Acc) and G-mean (G.M) are compared with ELM. Moreover, in Figure 4.1 which provides different structure based on number of neurons, it can be concluded that FLN has outperformed ELM from the perspective of all measures. In following Table 4.1 shown the comparison results are between FLN and ELM based standard evaluations.

Table 4-1 The Comparison result between ELM and FLN

No.Neurons	Model	MAX.Acc	AVR.Acc	DR	FAR	Precision	Recall	F.M	G.M
10	ELM	0.9255	0.8956	0.9047	0.1545	0.8956	0.8955	0.8955	0.8061
	FLN	**0.9641**	**0.9591**	**0.9586**	**0.0485**	**0.9588**	**0.9591**	**0.9587**	**0.9216**
25	ELM	0.9521	0.9471	0.9418	0.0695	0.9469	0.9472	0.9471	0.8977
	FLN	**0.9738**	**0.9669**	**0.9624**	**0.0441**	**0.9668**	**0.9666**	**0.9665**	**0.9367**
35	ELM	0.9631	0.9548	0.9501	0.0591	0.9632	0.9628	0.9629	0.9124
	FLN	**0.9785**	**0.9735**	**0.9696**	**0.0301**	**0.9781**	**0.9779**	**0.9783**	**0.9479**
50	ELM	0.9709	0.9652	0.9593	0.0478	0.9706	0.9701	0.9709	0.9321
	FLN	**0.9821**	**0.9803**	**0.9808**	**0.0247**	**0.9818**	**0.9822**	**0.9821**	**0.9606**

The table 4.1, shown the maximum (Best) and average accuracy (Mean), are computed for each algorithms (ELM, FLN), the experiments results taken as average for fifteen runs. The results of FLN based on a double parallel forward neural network, with this parallel connection of a multilayer feedforward neural network and a single layer feedforward neural network, and the DPFNN's output nodes not only receive the recodification of the external information through the hidden nodes, but also receive the external information itself directly through the input nodes.

This extra information will increase the learning rate of the model, which lead to make the FLN represented with less number of hidden neurons in hidden layer higher accuracy than the ELM as showed in Figure 4.1. Moreover, ELM shown higher false alarm rate in compare with FLN because of less number of weights in ELM in compare with FLN. In the following figures shown the comparisons between ELM and FLN with consideration for each part of number of neurons (10, 25, 35 and 50). Moreover, the average accuracy that FLN achieved better that ELM in all proposed different structures in this work with maximum accuracy 0.9821 achieved by FLN with 50 neurons in hidden layer. In both algorithms showed the impact of neurons in hidden layer based accuracy. Moreover, the FLN with only 10 neurons in the hidden layer got higher accuracy than ELM with 10,25 and 35 neurons in the hidden layer, which means achieved high accuracy with less complexity.

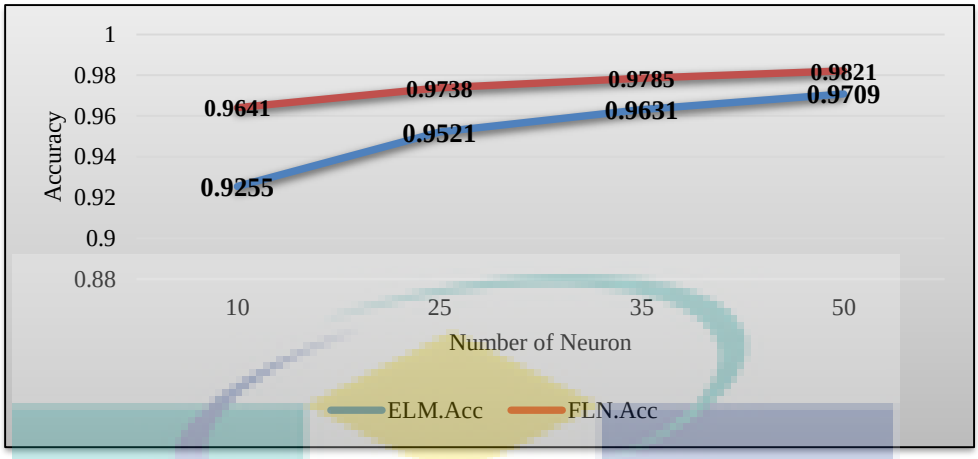


Figure 4.1 comparison of ELM vs FLN accuracy based number of neurons

In Figure 4.1, showed how the accuracy increase not in the same rate for both algorithms. The increase rate of accuracy is less in ELM algorithm because its start with low accuracy in compare with FLN in 10 hidden neurons which means based on the 2.6.1 section that represents the impact of double parallel forward neural network in FLN instead of single hidden layer in ELM algorithm. And even with 50 neurons the ELM accuracy didn't get equivalent FLN accuracy, which mean still need more hidden neurons to reach the same level with FLN accuracy.

4.3.2 Results of PSO-FLN

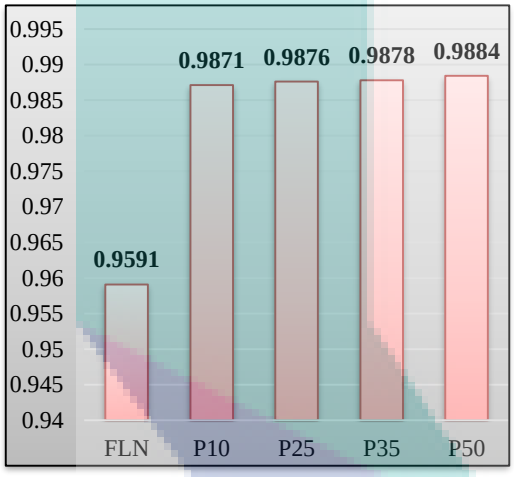
This section provides results of optimization models, as mentioned in previous chapter, the optimization's methods proposed to overcome the random selection of parameter's limitation in the basic FLN. The optimization PSO algorithm proposed to train basic FLN with a different number of iterations (100,250 and 500). Moreover, the results also analysis the effect of different particle swarm sizes 10, 25, 35 and 50 same as the different parts of the number of neurons in the hidden layer of FLN 10, 25, 35, and 50, to investigated the influence based on performance of the models. Table 4.2 shows the performance of PSO according to the training set with different number of neurons in the hidden layer and number of particles size and 500 maximum iterations; where P represent swarm size, M number of neurons and N.Itr is number of iterations. All the results in the following table represents as average for runs fifteen times.

Table 4.2 PSO-FLN Results

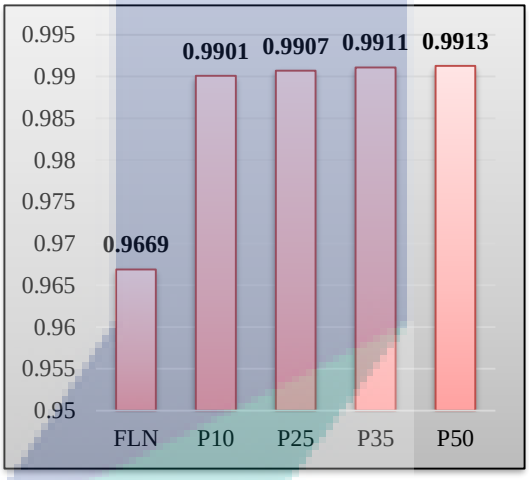
P	M	N.Itr	Best	Worst	Mean	Std.	DR	FAR	Precision	Recall	F.M	G.M
10	10	100	0.9892	0.9857	0.9871	0.0013	0.9709	0.0321	0.9716	0.9781	0.9751	0.9729
		250	0.9929	0.9879	0.988	0.0011	0.9746	0.0306	0.9736	0.9871	0.9769	0.9751
		500	0.9931	0.9871	0.9903	0.0008	0.9749	0.0291	0.9748	0.9817	0.9784	0.9768
	25	100	0.9928	0.9879	0.9901	0.0015	0.973	0.0314	0.9731	0.9871	0.9801	0.9787
		250	0.9931	0.9894	0.991	0.0012	0.9776	0.0272	0.9784	0.9843	0.9821	0.9804
		500	0.9938	0.9899	0.9913	0.0009	0.9803	0.0259	0.9804	0.9867	0.9823	0.9811
	35	100	0.9929	0.9897	0.9913	0.0009	0.9762	0.0411	0.9762	0.9886	0.9824	0.9812
		250	0.9952	0.9908	0.9925	0.0011	0.9807	0.0221	0.9808	0.9888	0.9847	0.9836
		500	0.9944	0.9906	0.9929	0.0010	0.9829	0.0249	0.9822	0.9889	0.9852	0.9846
	50	100	0.9945	0.9919	0.9932	0.0008	0.9831	0.0195	0.9831	0.9895	0.9865	0.9855
		250	0.9951	0.9939	0.9933	0.0007	0.9863	0.0156	0.9863	0.9894	0.9878	0.9871
		500	0.9955	0.9933	0.9936	0.0007	0.9877	0.0142	0.9876	0.9901	0.9892	0.9884
25	10	100	0.9895	0.9858	0.9876	0.0010	0.9711	0.058	0.9718	0.9799	0.9749	0.9731
		250	0.9905	0.9859	0.9888	0.0011	0.9741	0.03	0.9739	0.9791	0.9769	0.9749
		500	0.9924	0.9879	0.9905	0.0014	0.9795	0.0238	0.9793	0.9818	0.9807	0.9792
	25	100	0.9922	0.9887	0.9907	0.0010	0.9753	0.0291	0.9748	0.9864	0.9806	0.9792
		250	0.9942	0.9891	0.9915	0.0013	0.9799	0.0232	0.9797	0.9858	0.9828	0.9815
		500	0.9941	0.9905	0.9927	0.0010	0.9824	0.0203	0.9823	0.9881	0.9851	0.9841
	35	100	0.9933	0.9892	0.9919	0.0011	0.9801	0.0232	0.9799	0.9877	0.9837	0.9825
		250	0.9944	0.9921	0.9926	0.0007	0.9825	0.0202	0.9823	0.9892	0.9858	0.9851
		500	0.9961	0.9915	0.9932	0.0013	0.9836	0.0186	0.9837	0.9885	0.9861	0.9851
	50	100	0.9945	0.9924	0.9936	0.0006	0.9858	0.0168	0.9823	0.9887	0.9871	0.9862
		250	0.9944	0.9879	0.9939	0.0011	0.9831	0.0195	0.9827	0.9879	0.9856	0.9846
		500	0.996	0.9909	0.9942	0.0012	0.9904	0.0176	0.9905	0.9907	0.9905	0.9899
35	10	100	0.9892	0.9858	0.9878	0.0010	0.9715	0.0329	0.9713	0.9781	0.9747	0.9731
		250	0.9903	0.9868	0.9892	0.0010	0.9756	0.0286	0.9743	0.9804	0.9773	0.9755
		500	0.9905	0.9877	0.9912	0.0007	0.9755	0.0264	0.9753	0.9814	0.9783	0.9767
	25	100	0.9918	0.9881	0.9911	0.0013	0.9728	0.0316	0.9741	0.9861	0.9793	0.9778
		250	0.9939	0.9899	0.9919	0.0009	0.9787	0.0246	0.9786	0.9868	0.9833	0.9814
		500	0.9936	0.9926	0.9927	0.0007	0.9831	0.0197	0.9828	0.9872	0.9851	0.984
	35	100	0.9937	0.9907	0.9921	0.0008	0.9794	0.024	0.9792	0.9882	0.9834	0.9824
		250	0.9941	0.9911	0.993	0.0009	0.9819	0.0206	0.9822	0.9881	0.9849	0.9839
		500	0.9954	0.9926	0.9948	0.0008	0.9873	0.0146	0.9872	0.9899	0.9885	0.9877
	50	100	0.9948	0.9917	0.9939	0.0009	0.9844	0.0181	0.9842	0.9895	0.9868	0.986
		250	0.9961	0.9931	0.9945	0.0007	0.9877	0.0141	0.9876	0.9902	0.9889	0.9881
		500	0.9958	0.9935	0.995	0.0006	0.9894	0.0121	0.9893	0.9904	0.9899	0.9892
50	10	100	0.9905	0.9856	0.9884	0.0014	0.9706	0.0341	0.9705	0.9803	0.9753	0.9735
		250	0.9912	0.9875	0.9896	0.0010	0.9777	0.0257	0.9776	0.9791	0.9785	0.9769
		500	0.9928	0.9882	0.9915	0.0011	0.9781	0.0251	0.9776	0.9812	0.9794	0.9779
	25	100	0.9922	0.9895	0.99	0.0008	0.9753	0.0287	0.9751	0.9868	0.9809	0.9795
		250	0.9931	0.9901	0.9913	0.0009	0.9792	0.0240	0.9791	0.9885	0.9833	0.9817
		500	0.9951	0.9919	0.9931	0.0009	0.9853	0.0169	0.9852	0.9886	0.9861	0.9852
	35	100	0.9933	0.9906	0.9923	0.0008	0.9807	0.0224	0.9805	0.9876	0.9842	0.9829
		250	0.9946	0.9916	0.9935	0.0008	0.9834	0.0191	0.9833	0.9883	0.9857	0.9847
		500	0.9957	0.9933	0.9948	0.0007	0.9884	0.0132	0.9883	0.9897	0.989	0.9883
	50	**100**	**0.9949**	**0.9927**	**0.9941**	**0.0006**	**0.9849**	**0.0173**	**0.9848**	**0.9896**	**0.9872**	**0.9863**
		250	**0.9957**	**0.9933**	**0.9949**	**0.0006**	**0.9893**	**0.0124**	**0.9891**	**0.9896**	**0.9892**	**0.9884**
		500	**0.9964**	**0.9951**	**0.9959**	**0.0004**	**0.9933**	**0.0076**	**0.9932**	**0.9907**	**0.9921**	**0.9915**

Furthermore, Table 4.2 include several standard evaluations such as, number of iterations for each model, (Best) and (worst) accuracy of the fifteen runs for each models, (Mean) represents the average of accuracy. (Std) mean the standard deviation of the accuracy for each model. (DR) represents the detection rate and (FAR) is false alarm rate.

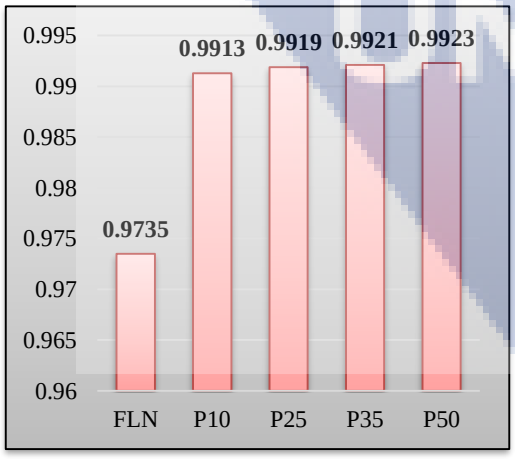
Moreover, the table shown the impact of number of neurons is more heavy than the number of particles and iteration numbers in accuracy of each models. In addition, the table shown PSO-FLN achieved better performance in compared to basic FLN because of the reduced the randomness of select main parameters impact in basic FLN. The following is the figures shown the comparison of accuracy between PSO-FLN with a difference's number of particles and basic-FLN.



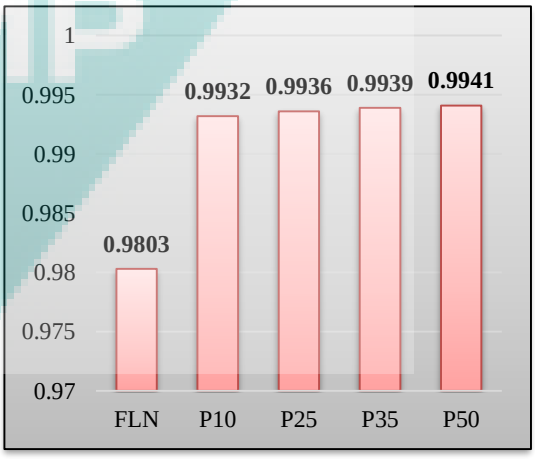
Figure(4.2)A. PSO-FLN Vs FLN with m=10 and Itr=100



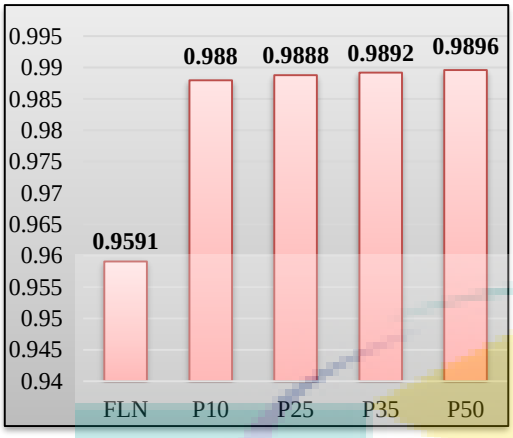
Figure(4.2)B. PSO-FLN Vs FLN with m=25 and Itr=100



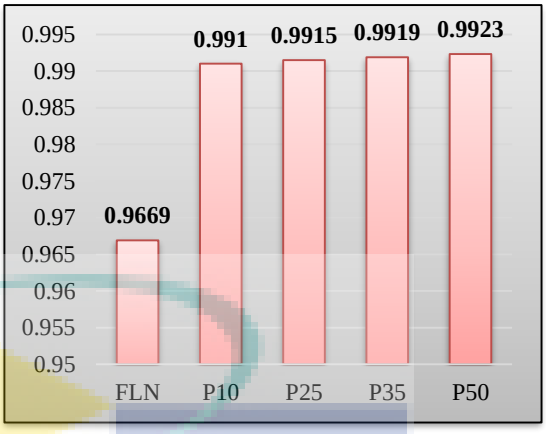
Figure(4.2)C. PSO-FLN Vs FLN with m=35 and Itr=100



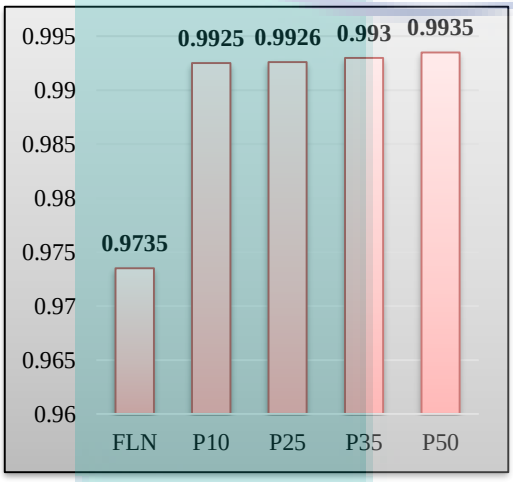
Figure(4.2)D. PSO-FLN Vs FLN with m=50 and Itr=100



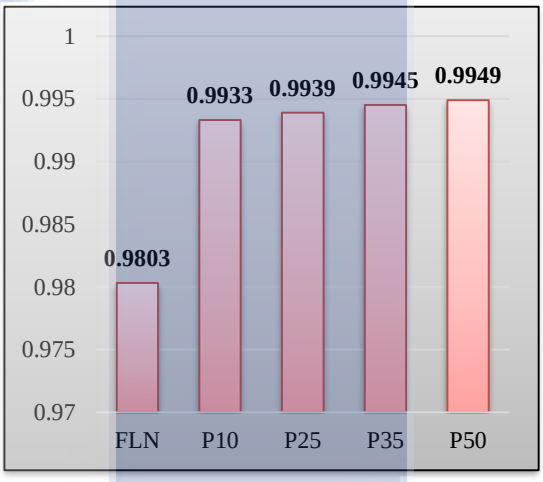
Figure(4.2)E. PSO-FLN Vs FLN with m=10 and Itr=250



Figure(4.2)F. PSO-FLN Vs FLN with m=25 and Itr=250



Figure(4.2)K. PSO-FLN Vs FLN with m=35 and Itr=250



Figure(4.2)O. PSO-FLN Vs FLN with m=50 and Itr=250

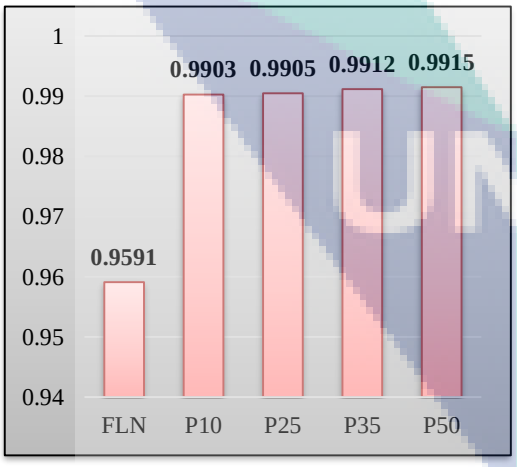


Figure (4..2)S. PSO-FLN Vs FLN with m=10 and Itr=500

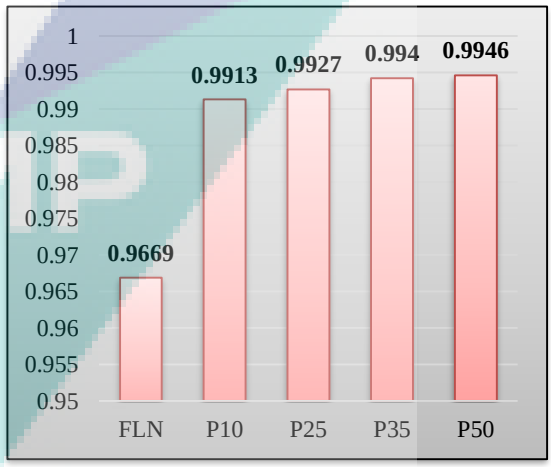


Figure (4.2)R. PSO-FLN Vs FLN with m=25 and Itr=500

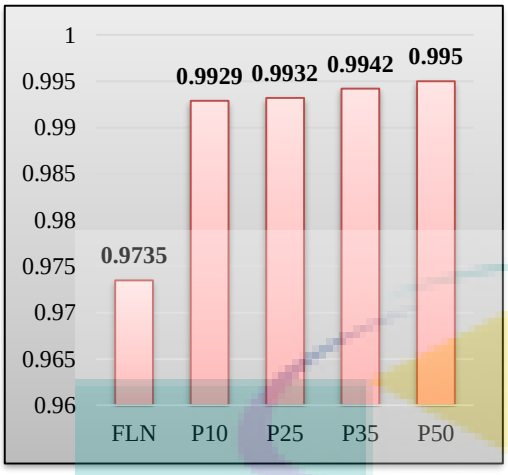


Figure (4.2) P.PSO-FLN Vs FLN with m=35 and Itr=500

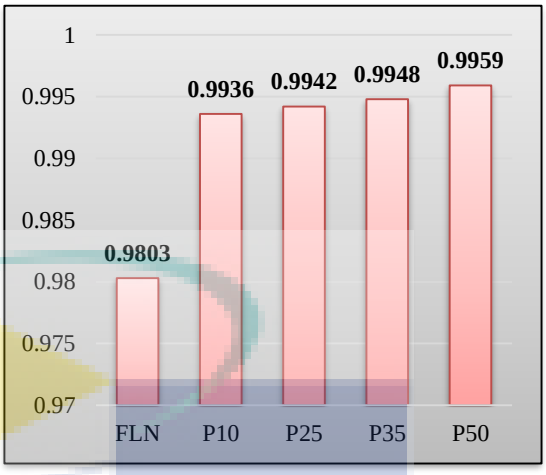


Figure (4.2) O.PSO-FLN Vs FLN with m=50 and Itr=500

Figure 4.2 Comparison between PSO-FLNs vs FLN Based accuracy

In Figure 4.2 represents PSO-FLN with different number of neurons (m) and different number of iterations (Itr) with different situations and (p) represents the number of particles. Moreover, all the cases A, B, C, D, E, F, K, O, S, R, P and O showed PSO-FLN achieved better results with all different structures in compare to the results of basic FLN. As a result of reduce the random selection of parameters in the basic FLN, PSO-FLN achieved better performance as shown in Table 4.2 and Figure 4.2. The accuracy with 50 neurons and 500 iterations achieved higher accuracy among all models of PSO-FLN

4.3.3 Result of MRPSO-FLN

This section provides the results of the new approach based PSO algorithm, (multi swarm optimization) as mentioned in previous chapter, the basic idea of this approach on update the position and velocity. On another hand, the parameters fixative during the update. This work proposed the idea of "Meeting Room Approach" to update the PSO parameters between several swarms based on the fitness.

Furthermore, MRPSO-FLN model divided into five clans with 10 members (particles) in each clan and 100 iterations, and its parameters (c_1, c_2,w) are updated based on steps of PSO algorithm. Once the new generation of each clan has been set, a new clan leader (the best leader) is elected and sent to the meeting room. The best among the leaders will be selected as the overall best leader (global best) in the meeting room based accuracy.

Table 4.3 PSO-FLN Vs MRPSO-FLN Results

Alg.	M	Best	Worst	Mean	Std.	DR	FAR	Precision	Recall	F.M	G.M
PSO-FLN	10	0.9928	0.9882	0.9915	0.0011	0.9777	0.0257	0.9776	0.9812	0.9794	0.9779
	25	0.995	0.9919	0.9931	0.0009	0.9853	0.0169	0.9852	0.9886	0.9861	0.9852
	35	0.9955	0.9933	0.9948	0.0007	0.9884	0.0132	0.9883	0.9897	0.989	0.9883
	50	0.996	0.9951	0.9957	0.0004	0.9933	0.0076	0.9932	0.9907	0.9921	0.9915
MRPSO-FLN	10	0.9931	0.9888	0.9921	0.0010	0.9786	0.0244	0.978	0.9815	0.9797	0.9789
	25	0.9951	0.9921	0.9935	0.0012	0.9862	0.0161	0.9858	0.9891	0.987	0.9855
	35	0.9957	0.9934	0.995	0.0009	0.9888	0.0128	0.989	0.9899	0.9896	0.9888
	50	0.9955	0.9948	0.9953	0.0007	0.9927	0.0080	0.9928	0.9899	0.9910	0.9906

In Table 4.3 the proposed model MRPSO-FLN compared with the best situation of 4.2 table, which contain PSO-FLN results, the results with 50 particles and 500 iterations in previous table achieved the better results in compared with other proposed models. On another hand, MRPSO-FLN is represented with the 50 particles divided into five clans (10 particles for clan) each of these clan's hybrid with a different number of neurons (10, 25, 35, and 50) in the hidden layer of the FLN with 100 iterations. The results for both models almost in the same accuracy range but with different structure such as the number of particles and iterations. Moreover, for 10, 25 and 35 neurons the PSO-FLN need for 500 iterations and 50 particles to achieve 0.9928, 0.995 and 0.9955 accuracy on other hand 0.9931, 0.9951 and 0.9957 accuracy for MRPSO with only 100 iterations and 10 particles, but with 50 neurons, PSO-FLN achieved 0.996 accuracy which slightly higher than 0.9955 for MRPSO-FLN. Moreover, in following a table 4.4 shown PSO parameter values which provides best accuracy based on different number of neurons.

Table 4.4 Parametervalues based on MRPSO-FLN

	M	Best	c_1	c_2	W
MRPSO-FLN	10	0.9931	1.4199	1.4195	0.7499
	25	0.995	1.4213	1.4209	0.7508
	35	0.9955	1.431	1.398	0.7498
	50	0.9955	1.4253	1.419	0.7422

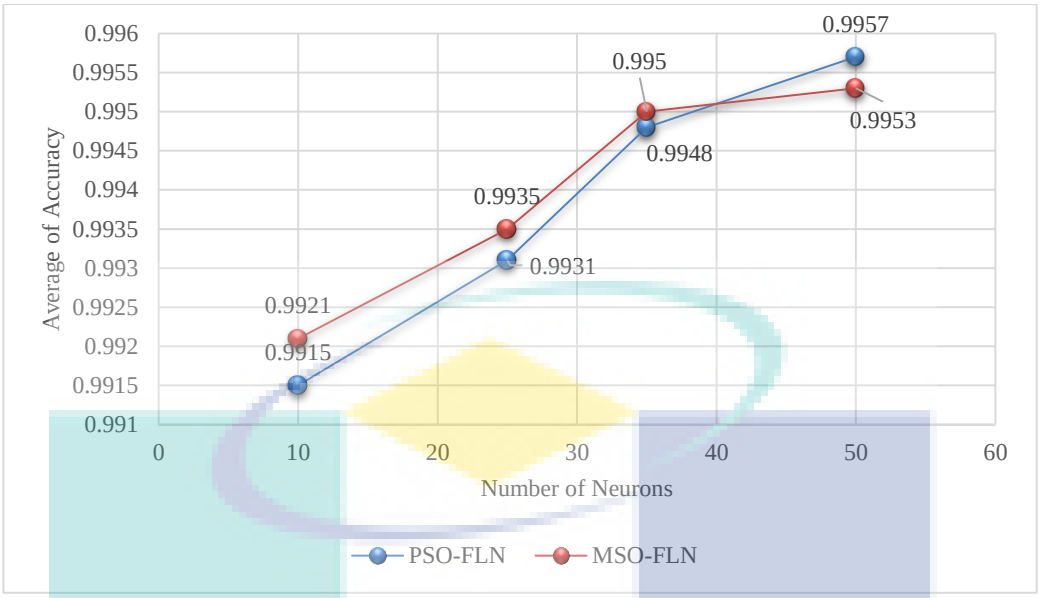


Figure 4.3 Comparison of average accuracy between PSO-FLN Vs MRPSO-FLN

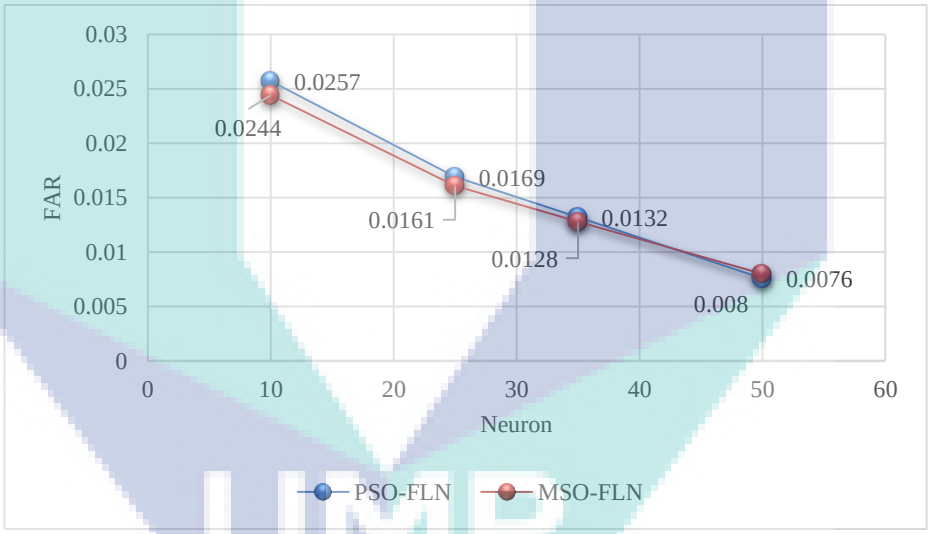


Figure 4.4 Comparison of average FAR between PSO-FLN Vs MRPSO-FLN

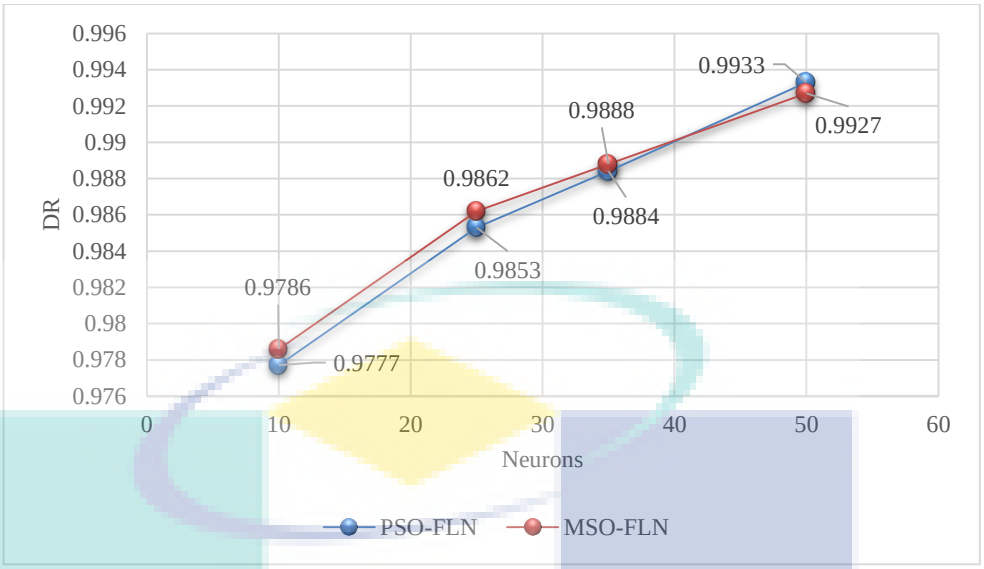


Figure 4.5 Comparison of average DR between PSO-FLN Vs MRPSO-FLN

Figure 4.3 showed the accuracy for both PSO-FLN and MR PSO-FLN, only with 50 neutrons MR PSO-FLN achieved worse than PSO-FLN which because the impact of structure differentiations. The results between the PSO-FLN and MRPSO-FLN are convergent, but as mentioned before the PSO-FLN adjusts with 500 iterations and 50 swarm particles, on another hand MRPSO-FLN only with 100 iterations and 10 swarm particles. Which mean less in the complexity and time than PSO-FLN without high losing at the performance rate.

4.4 The Comparative of Proposed Models

To illustrate the effectiveness of our proposed IDS models, this section represents a compare proposed models performance with 15 recently developed anomaly detection techniques. Table 4.5 demonstrates the result achieved by proposed models compared with other models tested on NSL-KDD dataset in term of detection rate and false alarm rate. It is very clear that our proposed models (PSO-FLN, MRPSO-FLN) gets the best results with 0.9933 detection rate, 0.0076 false alarm rate for PSO-FLN model as mentioned in table 4.2, and 0.9927 detection rate, 0.0080 false alarm rate for MRPSO-FLN model, in following Table 4.5

Table 4-4 comparative of proposed models

Ref	Algorithms Used	P.S	DR	FAR
(Hajimirza& Navimipour, 2018)	Proposed a combination of a MLP and Artificial bee Colony (ABC) and k-means clustering	---	0.9841	0.012
(Gauthama Raman et al., 2017)	Proposed HG-GASVM based IDS which using hypergraph - genetic algorithm for parameter setting in support SVM	C $2^{-5}-2^{5}$,γ= $2^{-4}-2^{4}$,Crossover rate=80% Mutation rate=2% Iterations =500	0.9714	0.83
(Hosseini Bamakan et al., 2016)	Proposed time varying chaos particle swarm optimization(TVCPSO) to parameters setting SVM	C $2^{-5}-2^{5}$,γ= $2^{-4}-2^{4}$,$\alpha=-10^{-5}-10^{-1}$ Iterations=200,S.S =8	0.970	0.87
(Singh et al., 2015)	Proposed IDS based on traffic profiling and online sequential extreme learning machine (OS-ELM)	M=50--1000 Threshold=0.5-2.0	0.9767	0.017
(Eduardo et al., 2013)	Proposed IDS based on hybrid SVM classifier and Non-linear projection technique	α=5 d=3	0.9341	0.14
(Ahmad & Senga, 2017)	Proposed IDS based on SVM and BAT algorithm	-----	0.9431	0.5
(Enache & Sgarciu, 2014)	Proposed IDS based on hybrid PSO and SVM	γ=0.1α= 0.9,C1=2.3 C2=1.8,W=0.9to0.5, A_0 =1	0.9341	0.049
	Proposed IDS based on hybrid Bat and SVM	0,r_o=0.9,FR=0.8 to 1.0	0.956	0.04
(AL-Hawawreh et al., 2018)	Proposed IDS by using a deep auto- encoder and deep ANN	Epochs=100,L1=L2=1e^{-6} ,M=10-3-10 AR=2e^{-6},Ramp=1e^{7}	0.99	0.01
(Ambusaidi et al., 2016)	Proposed IDS based on hybrid Least Square Support Vector Machine based IDS (LSSVM-IDS)	β =1-0, Step size =0.1 K=6	0.9596	0.38
(Dinh & Ngoc, 2017)	Proposed IDS based on hybrid a stacked auto encoder and random forest named (SAE-RF)	M=100 B=0-2	0.8542	0.036
(Ludwig, 2017)	Proposed IDS based on neural network ensemble	A. S=10—20 N. N=15-20 Layers=2 M=20-25	0.925	0.147
(Hajisalem & Babaie, 2018)	Proposed IDS used a hybrid classification based on Artificial Bee Colony (ABC) and Artificial Fish Swarm (AFS)	SN=30, Food Number=15 Limit=5000, Runtime=10 U=500, L=-500,D=1500	0.990	0.01
(Ambusaidi et al., 2016)	Proposed IDS based on a Filter-Support Vector Machine (FSVM)	β =0.3,1, Iterations=10 K=6	0.9229	0.02
(Ji et al., 2016)	Proposed IDS based on a Multi-Level DWT	K=3, Normalization=0 -1 γ =0.99	0.9123	0.023
(Moustafa et al., 2017)	Proposed IDS based on a Geometric Analysis (GAA)	(π, υ, ω) = (0.55,30,10) (π, υ, ω)= (0.45, 10,30)	0.981	0.002
Proposed Models	**Proposed IDS based on Basic-FLN**	**M=50**	**0.9808**	**0.024**
	Proposed IDS based on PSO-FLN	**M=50, Iteration =500 S.S=50**	**0.9933**	**0.007**
	Proposed IDs based on MRPSO-FLN	**M=50, Iteration=100 S.S=10**	**0.9927**	**0.0080**

* r_o: MINIMUM PULSE RATE	*M: NEURONS IN HIDDEN LAYER	*K: K-NEAREST SET
* A_0: MAXIMUM LOUDNESS	*C, Γ: SVM PARAMETER	*A. S: AUTO ENCODER SIZE
*D: RBF KERNEL PARAMETERS	*α:POLYNOMIAL PARAMETER	*N. N: NUMBER OF NODES
*S. S: SWARM SIZES	*L: LOWER BOUNDARIES	*U: UPPER BOUNDARIES
*AR: ANNEALING RATE	*L1, L2: REGULARIZATION	*B: CONFUSED PARAMETER
*x_1, x_2 : ARBITRARY VARIABLES	* β: CONTROL PARAMETER	*SN: FOOD SOURCES

Table 4.5 structure contains five columns, start with first column that represents references. The second column represents the main methods that proposed as IDS. Moreover, third column represents the parameters setting (P.S) of the main methods. The fourth column in table represents detection rate (DR), and final column represents false alarm rate (FAR). All the proposed models in the above table share NSL-KDD data set with multi classification based on anomaly IDS. The results shown that there is a significant impact of FLN structure models based on IDS performance.

Furthermore, the improvement of selected FLN main parameters based PSO algorithms are also shown in the table. Moreover, this section represents the statistical analysis for the proposed models based on accuracy comparison. The statistical test typically used for preferable comparison, for this proposed by using the results obtained for 15 runs for each models (Francisco J. Samaniego, 2014). In this work a Wilcoxon signed rank test is performed with a statistical significance value $\propto$=0.05. The null hypothesis H_0 for this test " There is no different between the median of the solutions produced by algorithm A and the median of solutions produced by algorithm B for the same benchmark problem". To determine whether algorithm A reached a statistically better solution than (B), or if not, whether the alternative hypothesis is valid, the size of the ranks provided by the Wilcoxon signed rank test (T+, T-) are examined. Table 4.6 showed the statistical pairwise results. Firstly, the basic FLN compared to ELM algorithm. Secondly, PSO-FLN compared to FLN.

Table 4.5 The Wilcoxon Signed Rank Test

MODELS	+	-	Z.V	P.V	RESULT
ELM Vs FLN (m=10)	15	0	-3.397	0.000979	H_o Reject
ELM Vs FLN (m=25)	15	0	-3.296	0.000982	H_o Reject
ELM Vs FLN (m=35)	15	0	-3.408	0.000655	H_o Reject
ELM Vs FLN (m=50)	15	0	-3.397	0.000973	H_o Reject
FLN (m=10) Vs PSO-FLN(m=10, Number of particles=10)					
Number of Iterations=100	15	0	-3.297	0.000652	H_o Reject
Number of Iterations=250	15	0	-3.409	0.000655	H_o Reject
Number of Iterations=500	15	0	-3.410	0.000650	H_o Reject
FLN (m=10) Vs PSO-FLN(m=10, Number of particles=25)					
Number of Iterations=100	15	0	-3.408	0.000653	H_o Reject
Number of Iterations=250	15	0	-3.405	0.000655	H_o Reject
Number of Iterations=500	15	0	-3.297	0.000979	H_o Reject
FLN (m=10) Vs PSO-FLN(m=10, Number of particles=35)					
Number of Iterations=100	15	0	-3.297	0.000979	H_o Reject
Number of Iterations=250	15	0	-3.408	0.000653	H_o Reject
Number of Iterations=500	15	0	-3.413	0.000642	H_o Reject
FLN (m=10) Vs PSO-FLN(m=10, Number of particles=50)					
Number of Iterations=100	15	0	-3.409	0.000652	H_o Reject
Number of Iterations=250	15	0	-3.517	0.000437	H_o Reject
Number of Iterations=500	15	0	-3.517	0.000436	H_o Reject
FLN (m=25) Vs PSO-FLN(m=25, Number of particles=10)					
Number of Iterations=100	15	0	-3.409	0.000652	H_o Reject
Number of Iterations=250	15	0	-3.408	0.000655	H_o Reject
Number of Iterations=500	15	0	-3.409	0.000653	H_o Reject
FLN (m=25) Vs PSO-FLN(m=25, Number of particles=25)					
Number of Iterations=100	15	0	-3.408	0.000652	H_o Reject
Number of Iterations=250	15	0	-3.409	0.000653	H_o Reject
Number of Iterations=500	15	0	-3.408	0.000655	H_o Reject
FLN (m=25) Vs PSO-FLN(m=25, Number of particles=35)					
Number of Iterations=100	15	0	-3.408	0.000653	H_o Reject
Number of Iterations=250	15	0	-3.409	0.000653	H_o Reject
Number of Iterations=500	15	0	-3.408	0.000651	H_o Reject
FLN (m=25) Vs PSO-FLN(m=25, Number of particles=50)					
Number of Iterations=100	15	0	-3.409	0.000655	H_o Reject
Number of Iterations=250	15	0	-3.411	0.000648	H_o Reject
Number of Iterations=500	15	0	-3.409	0.000652	H_o Reject

Table 4.5 continue

FLN (m=35) Vs PSO-FLN(m=35, Number of particles=10)					
Number of Iterations=100	15	0	-3.408	0.000655	H_o Reject
Number of Iterations=250	15	0	-3.408	0.000653	H_o Reject
Number of Iterations=500	15	0	-3.409	0.000651	H_o Reject
FLN (m=35) Vs PSO-FLN(m=35, Number of particles=25)					
Number of Iterations=100	15	0	-3.411	0.000653	H_o Reject
Number of Iterations=250	15	0	-3.408	0.000655	H_o Reject
Number of Iterations=500	15	0	-3.409	0.000653	H_o Reject
FLN (m=35) Vs PSO-FLN(m=35, Number of particles=35)					
Number of Iterations=100	15	0	-3.408	0.000651	H_o Reject
Number of Iterations=250	15	0	-3.409	0.000655	H_o Reject
Number of Iterations=500	15	0	-3.410	0.000650	H_o Reject
FLN (m=35) Vs PSO-FLN(m=35, Number of particles=50)					
Number of Iterations=100	15	0	-3.408	0.000655	H_o Reject
Number of Iterations=250	15	0	-3.409	0.000653	H_o Reject
Number of Iterations=500	15	0	-3.408	0.000655	H_o Reject
FLN (m=50) Vs PSO-FLN(m=50, Number of particles=10)					
Number of Iterations=100	15	0	-3.409	0.000653	H_o Reject
Number of Iterations=250	15	0	-3.408	0.000655	H_o Reject
Number of Iterations=500	15	0	-3.409	0.000652	H_o Reject
FLN (m=50) Vs PSO-FLN(m=50, Number of particles=25)					
Number of Iterations=100	15	0	-3.408	0.000652	H_o Reject
Number of Iterations=250	15	0	-3.409	0.000653	H_o Reject
Number of Iterations=500	15	0	-3.408	0.000655	H_o Reject
FLN (m=50) Vs PSO-FLN(m=50, Number of particles=35)					
Number of Iterations=100	15	0	-3.409	0.000655	H_o Reject
Number of Iterations=250	15	0	-3.408	0.000653	H_o Reject
Number of Iterations=500	15	0	-3.412	0.000645	H_o Reject
FLN (m=50) Vs PSO-FLN(m=50, Number of particles=50)					
Number of Iterations=100	15	0	-3.411	0.000647	H_o Reject
Number of Iterations=250	15	0	-3.409	0.000652	H_o Reject
Number of Iterations=500	15	0	-3.408	0.000651	H_o Reject

In this table, + indicates the positive ranks, while – indicated the negative ranks. In the z-distribution column, and when the p-value of less than 0.05, which means there is a significant difference between the two algorithms in that test. The table can be summarized as follows. In FLN vs ELM part, the test indicates that there are more significant negative ranks(N=15) without significant positive ranks based on all proposed different algorithm's structure (number of neurons). This means that the median of FLN is more than a median of ELM. In other words, H_0 is rejected and the FLN has better performance and has outperformed ELM. Moreover, in PSO-FLN, the test indicates that the all results are significant negative also without significant positive ranks. This means H_0 is rejected and the PSO-FLN has better performance and has outperformed FLN.

4.5 Validate of The Proposed Methods

This section provides a more detail to validate the proposed models, in general this part divided into two parts. The first part, provides a validate proposes to PSO-

FLN. The second part, propose a validate of basic FLN. In general, these validation works based on the information form the MRPSO-FLN model.

4.5.1 Validate of PSO-FLN

In previous section, the experiments of PSO-FLN adjusted the parameters based on (Xia et al., 2018) work, which explained a newest and poplar of multi swarm based on PSO based on purposeful detecting. on another hand, in the end of MRPSO-FLN experiments, the model provides the best c_1, c_2 and w based on room meeting approach. The propose of this section is to provides the parameters value that got from MRPSO-FLN as the best accuracy provided with parameters value c_1=1.431, c_2=1.395,w=0.749 for 35 neurons, and applied in the PSO-FLN Instead of the standard value of parameters that had been used in previous PSO-FLN model.

Table 4.6 PSO-FLN* Vs PSO-FLN(p=10) Results

Alg.	M	Best	Worst	Mean	Std.	DR	FAR	Precision	Recall	F.M	G.M
PSO-FLN	10	0.9892	0.9857	0.9871	0.0013	0.9709	0.3214	0.9716	0.9781	0.9751	0.9729
	25	0.9928	0.9879	0.9901	0.0015	0.973	0.0314	0.9731	0.9871	0.9801	0.9787
	35	0.9929	0.9897	0.9913	0.0009	0.9762	0.0411	0.9762	0.9886	0.9824	0.9812
	50	0.9945	0.9919	0.9932	0.0008	0.9831	0.0195	0.9831	0.9895	0.9865	0.9855
PSO-FLN*	10	0.9931	0.9888	0.992	0.001	0.9786	0.0247	0.978	0.9815	0.9797	0.9789
	25	0.995	0.9922	0.9935	0.0012	0.9862	0.017	0.9858	0.9891	0.987	0.9855
	35	0.9955	0.9934	0.995	0.0009	0.9888	0.0131	0.989	0.9899	0.9896	0.9888
	50	0.9958	0.9948	0.9953	0.0007	0.9927	0.0084	0.9928	0.9899	0.991	0.9906

The table 4.7 showed the number of iterations for both of models 100 and 10 swarm particles, which mean same structures. The performance of PSO-FLN* which represents the PSO-FLN with parameters value that provided by MRPSO-FLN, achieved better performance than the PSO-FLN with the standard parameter's values.

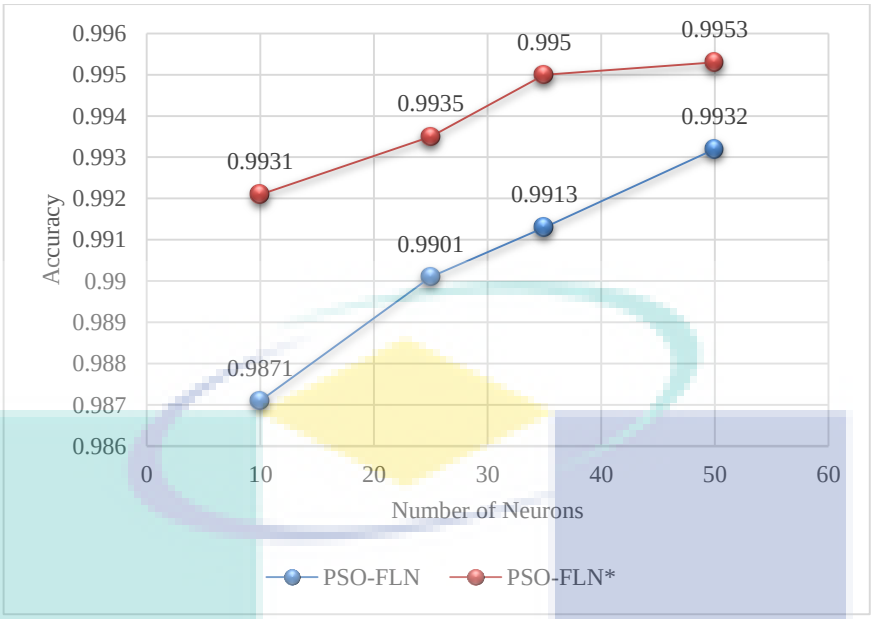


Figure 4.6 Comparison of Accuracy between PSO-FLN (P=10) Vs PSO-FLN*

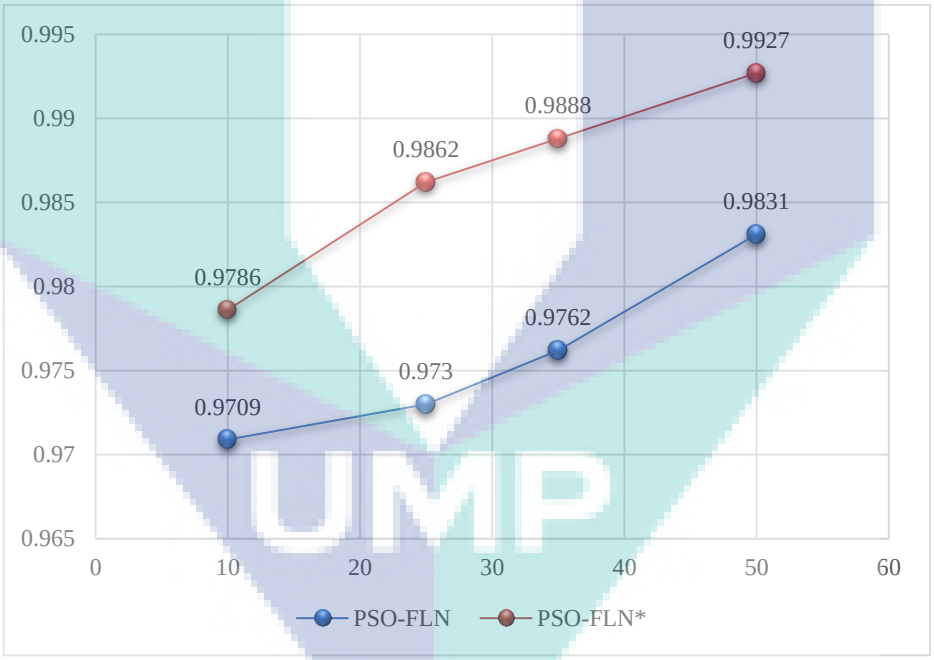


Figure 4.7 Comparison of average DR between PSO-FLN(P=10) Vs PSO-FLN*

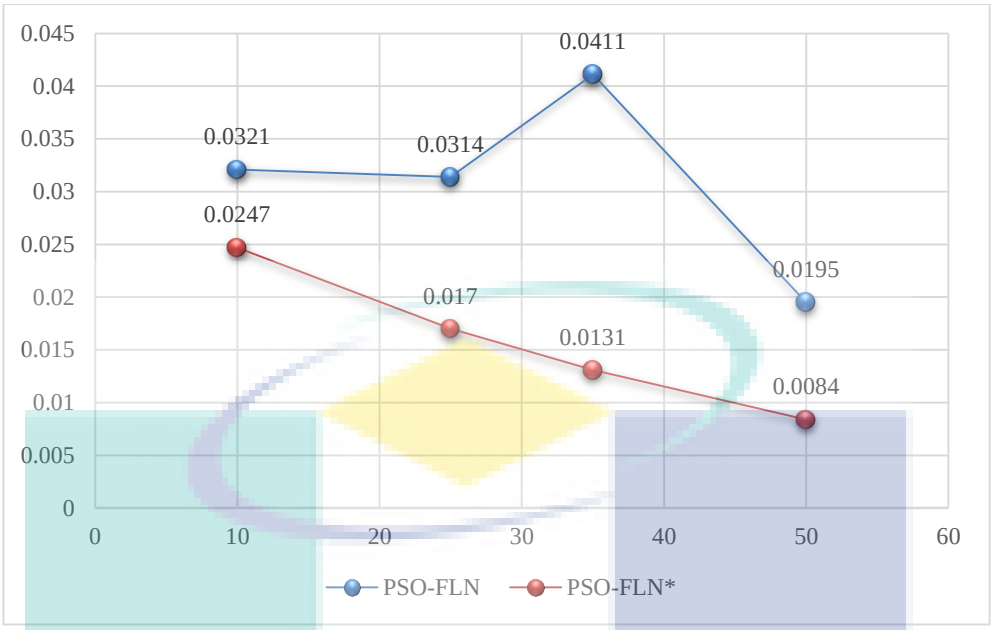


Figure 4.8 Comparison of average FAR between PSO-FLN) P=10) Vs PSO-FLN*

All the above figures in general showed that PSO-FLN* is based the new values of c_1, c_2 and w achieved better performance in compare with FLN-PSO based the standard parameters values. Figure 4.6 represents the comparison between PSO-FLN and PSO-FLN* based on average accuracy. the PSO-FLN* with only 10 neurons the average of PSO-FLN* accuracy was better than the average accuracy of standard PSO-FLN even with 25 and 35 neurons in the hidden layer. As a conclusion, the PSO-FLN* with parameters value that provided from MRPSO-FLN methods with fewer numbers of neurons in the hidden layer achieved better accuracy and detection rate as in Figure 4.7 with less complexity. Moreover, Figure 4.8 represents the comparison between PSO-FLN and PSO-FLN* is based on false alarm rate. In general, the performance PSO-FLN* is better and achieved less rate in all structure of hidden layer. Moreover, the decrease of FAR was regularly with the increase of the number of neurons in PSO-FLN*, when the decrease of FAR was not regularly in PSO-FLN as shown in figure during 35 neurons in hidden layer. Moreover, this section analysis the PSO-FLN* comparison results with best results based on Table 4.2, PSO-FLN model with 50 particles better than other in general. As following, Table 4.8 shows the comparison between the standard PSO-FLN and PSO-FLN* with different structures as 50 particles and 500 iterations for PSO-LN and10 particles and 100 iterations for PSO-FLN*.

Table 4.7 PSO-FLN* Vs PSO-FLN(p=50) Results

Alg.	M	Best	Worst	Mean	Std.	DR	FAR	Precision	Recall	F.M	G.M
PSO-FLN	10	0.9928	0.9882	0.9915	0.0011	0.9777	0.0257	0.9776	0.9812	0.9794	0.9779
	25	0.9951	0.9919	0.9931	0.0009	0.9853	0.0169	0.9852	0.9886	0.9861	0.9852
	35	0.9957	0.9933	0.9948	0.0007	0.9884	0.0132	0.9883	0.9897	0.989	0.9883
	50	0.9959	0.9951	0.9959	0.0004	0.9933	0.0076	0.9932	0.9907	0.9921	0.9915
PSO-FLN*	10	0.9931	0.9888	0.9921	0.001	0.9786	0.0247	0.978	0.9815	0.9797	0.9789
	25	0.995	0.9921	0.9935	0.0012	0.9862	0.017	0.9858	0.9891	0.987	0.9855
	35	0.9955	0.9934	0.995	0.0009	0.9888	0.0131	0.989	0.9899	0.9896	0.9888
	50	0.9958	0.9948	0.9953	0.0007	0.9927	0.0084	0.9928	0.9899	0.991	0.9906

In table 4.8, the main differences between the models that have been compared are the numbers of particles and iterations. Moreover, the results of the models were slightly different as explains in following figures.

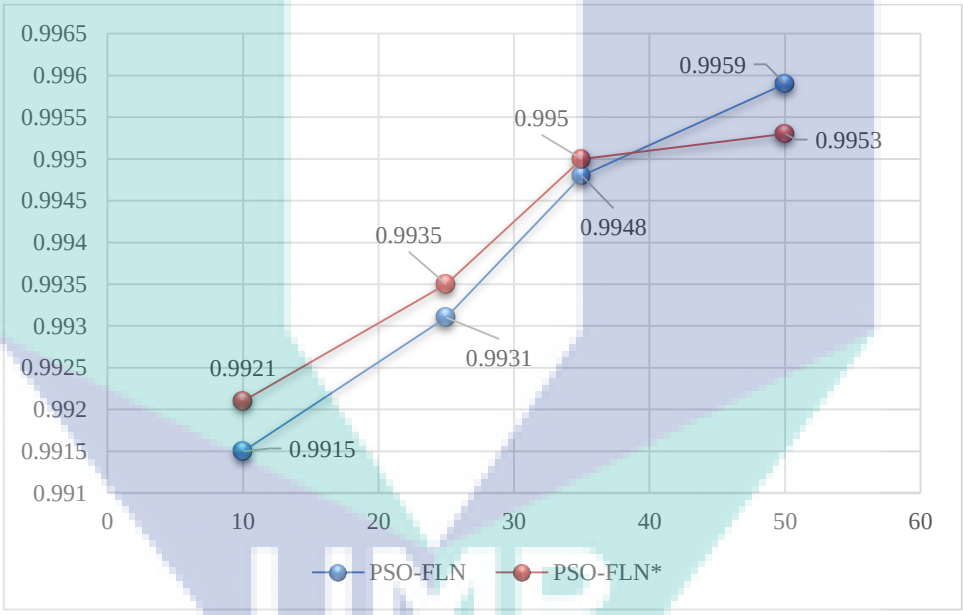


Figure 4.9 Comparison of Accuracy between PSO-FLN (P=**50**) Vs PSO-FLN*

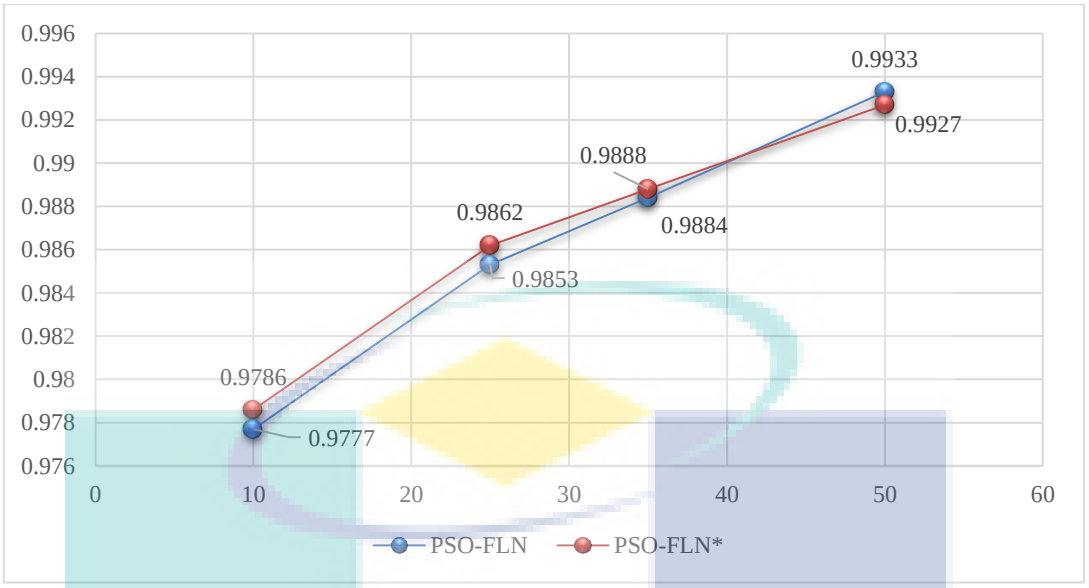


Figure 4.10 Comparison of average DR between PSO-FLN(P=**50**) Vs PSO-FLN*

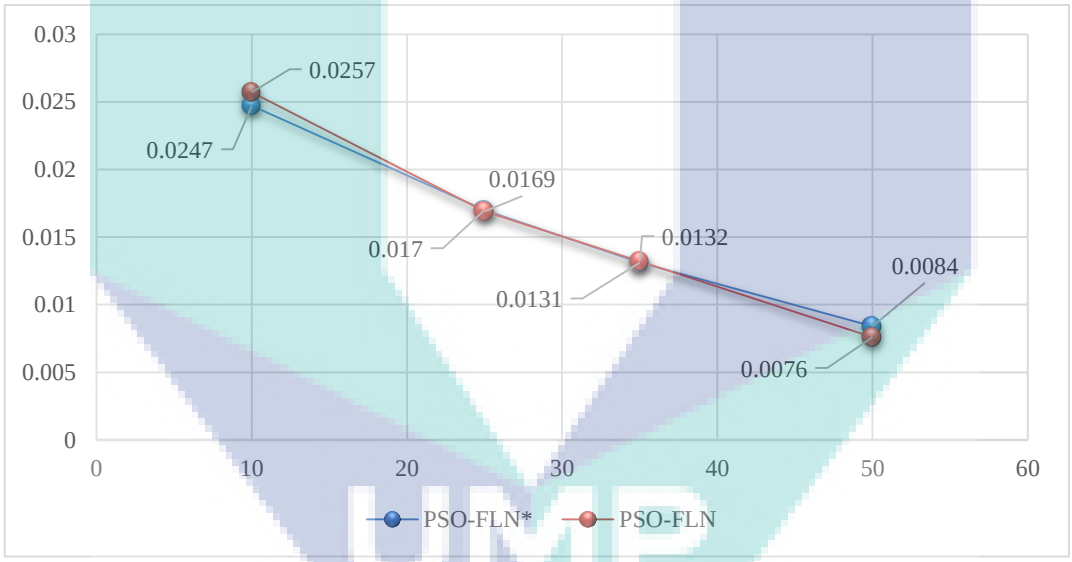


Figure 4.11 Comparison of average FAR between PSO-FLN(P=**50**) Vs PSO-FLN*

Furthermore, Figures 4.9 and 4.10 represented the comparison based on the differences of average accuracy and detection rate. The proposed of PSO-FLN* model achieved slightly higher accuracy than the PSO-FLN during (10, 25, 35) neurons in the hidden layer, and the standard PSO-FLN achieved higher accuracy with 50 neurons in the hidden layer. Faintly, false alarm rate's comparison showed in the Figure 4.11.

4.5.2 Validate of FLN

This section provides a valid set to basic FLN, as mentioned in previous sections that FLN selected main parameters randomly. Moreover, this section proposed to provide FLN parameters based takes from PSO-FLN* instead of create randomly. In the end of PSO-FLN* processed which its work based on best PSO parameters as mentioned in above section, can get the best values of FLN parameters. These parameters represent in FLN as w_i and b_i.Moreover, with specification's PC for experiments implementations include. Operating system is 64 bit windows 10 Pro, with processor core i7 and 16.0 GB memory. In another hand, the official NSL-KDD provided by websites has a total number of 24 attack types, and based, attack categories can be classified into 5 categories in order to compression with new model free parameters FLN. Table 4.9 as following presents the parameters setting for all the benchmark algorithm used in this study.

Table 4.8 Parameters setting

Algorithm	Parameter	Value
Decision Tree((Hoeffiding)	Grace Period	200
	Batch Size	100
	Hoeffding theeshold	0.05
Naïve Bayes	Batch Size	100
	Num Decimal Places	2
MLP(back propagation)	Batch Size	100
	Learning Rate	0.3
	Training Time	500
	Num Decimal Places	2
	momentum	0.2
	Num of Neurons	10
SVM	Batch Size	100
	C	0.1
	Num Folds	-1
	Tolerance Parameter	0.001
ELM	Num of Neurons	10
Basic FLN	Num of Neurons	10

Table 4.9 represents the parameter's values setting of the benchmark algorithms. In addition to the new FLN free parameters model, this section provides. It's also provides a compare of the new model with several standard models include (basic FLN, ELM, SVM, Naïve Bayes, Multilayer Perceptron (MLP) and Hoeffding tree (Decision tree)) based on NSL-KDD. Weka version 3.8.2 is used in this work to

Implementation of benchmark models based on NSL-KDD dataset. Moreover, the setting of free parameters Improved FLN (IFLN), which proposed in this section also adjust with 10 neurons in the hidden layer. As following Table 4.10 shown comparison results.

Table 4.9 comparison results of Improved FLN

Model	Accuracy	DR	FAR	Precision	Recall	F.M	Time(s)
Hoeffiding	0.9636	0.964	0.052	0.970	0.964	0.966	7.64
NaïveBayes	0.8357	0.836	0.044	0.914	0.836	0.971	1.48
MLP	0.9811	0.9812	0.021	0.9822	0.9834	0.9821	4227
SVM	0.9831	0.9822	0.017	0.9833	0.983	0.9831	245.9
ELM	0.9255	0.9047	0.1545	0.8956	0.8955	0.9855	0.448
Basic FLN	0.9641	0.9586	0.0485	0.9588	0.9591	0.9587	0.485
IFLN	0.9908	0.9825	0.014	0.9707	0.9822	0.9755	0.422

In Table 4.7, the results shown the IFLN model has better accuracy in compare with other models. Furthermore, the false alarm rate for IFLN is less than most of the models in the table. In following figures 4.12, 4.13 and 4.14 are shown

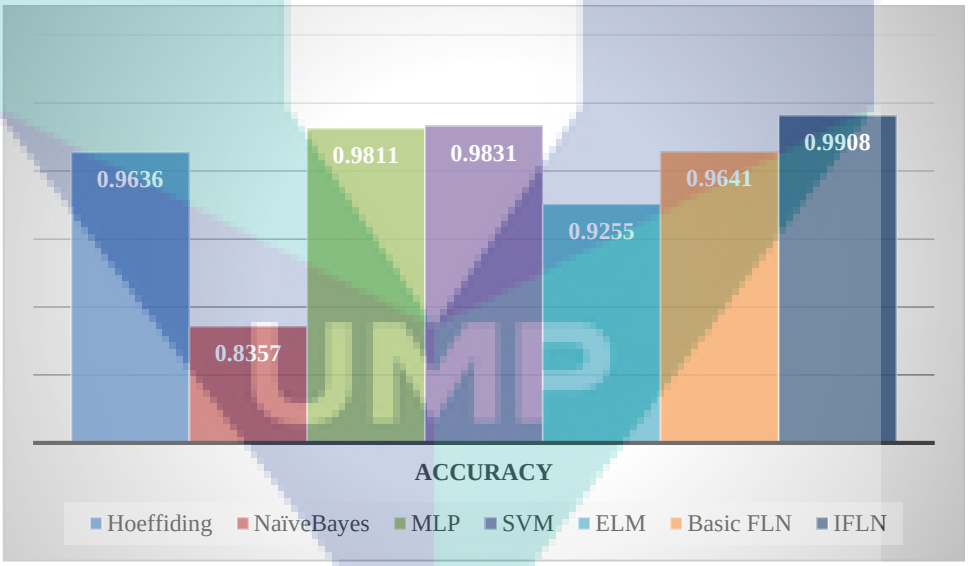


Figure 4.12 Accuracy comparison of IFLN

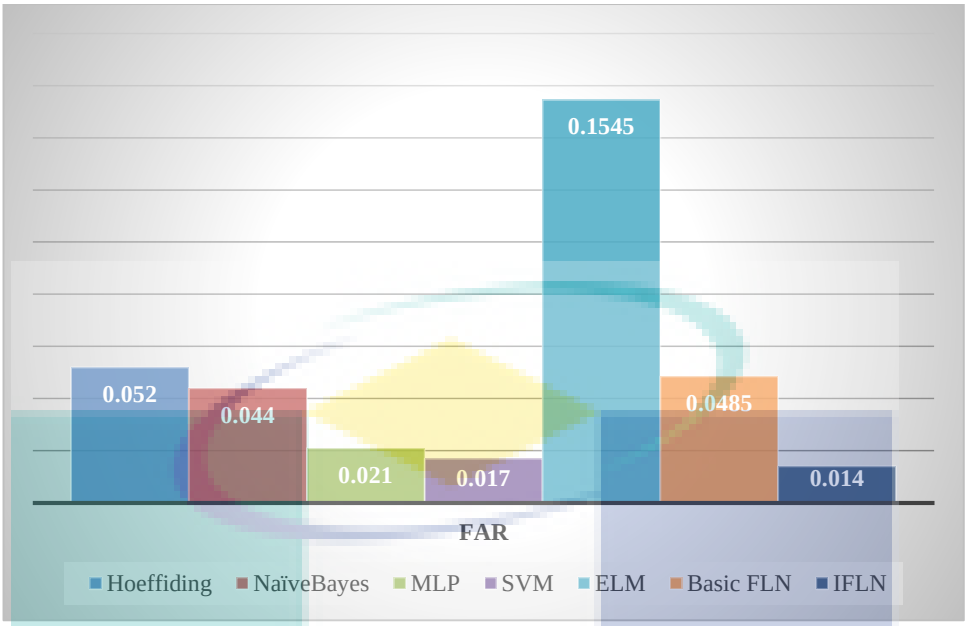


Figure 4.13 False Alarms comparison of IFLN

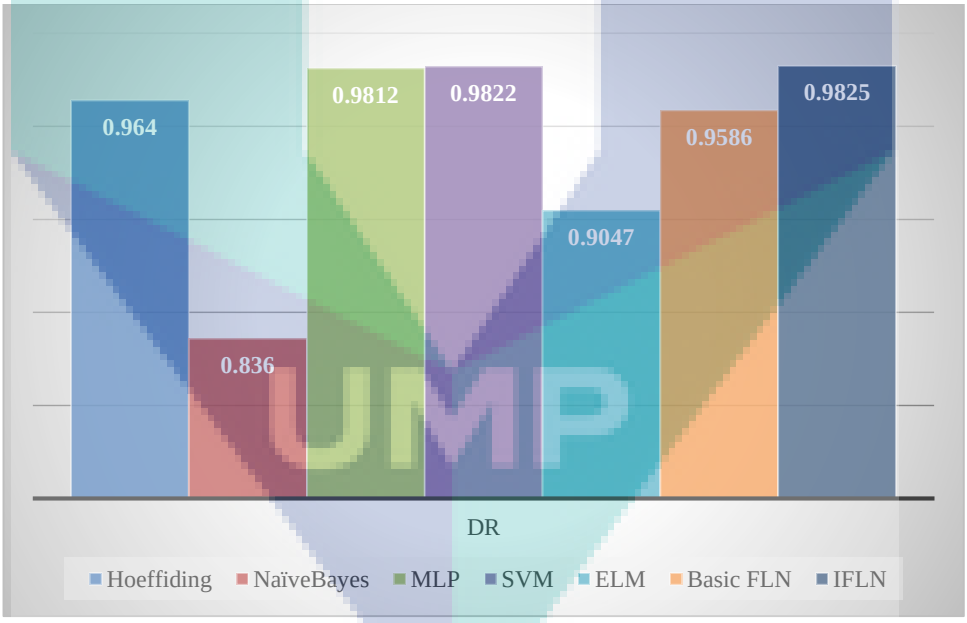


Figure 4.14 Detection Rate comparison of IFLN

4.6 DISCUSSION

Most studies on network intrusion detection only supply the overall detection accuracy rate, and few supply the false alarm rate, detection rate. Moreover, this chapter in general contains 4 sections based on the proposed of previous chapter.

Furthermore, all models proposed in this work evaluated based NSL-KDD dataset, which contains 148517 records with 42 attributes.

Firstly, proposed FLN based on network intrusion detection and model results compared with ELM as evaluated as shown in table 4.1. Additionally, the table shows the different neurons number in the hidden layer for both algorithms to investigate the influence of the neuron's increase on the performance, which represents a different algorithm structure. The FLN results showed more stability and better accuracy than ELM algorithm.

As a result of the differences in algorithm's structure, the FLN structure contains is a double parallel forward neural network which makes output nodes not only receive information from the hidden nodes, but also get the external information directly from the input nodes. On other hands, the ELM structure contains single Hidden Layer Feedforward Neural Network, which make the output nodes receive the information from hidden nodes.

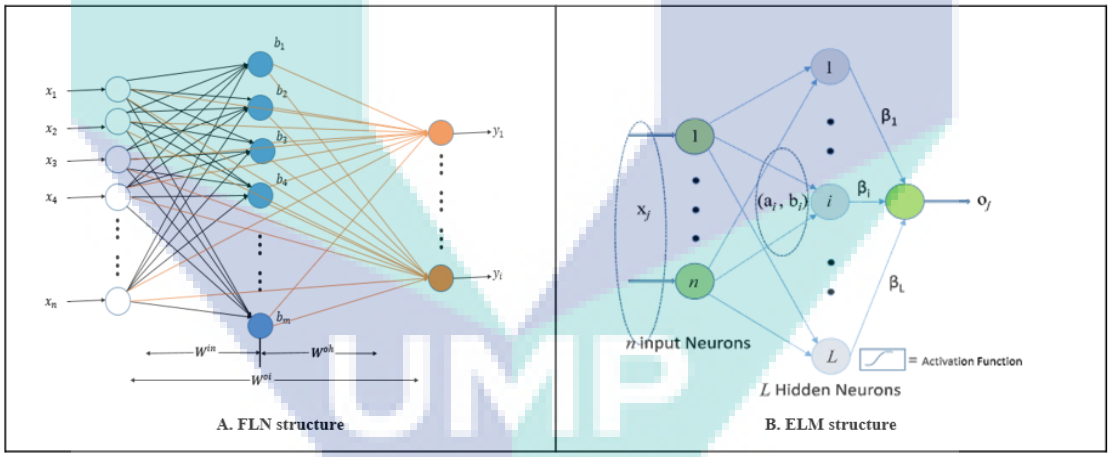


Figure 4.15 structure differences

Furthermore, Table 4.1 showed that ELM needs 50 neurons to reach accuracy 0.9709, when the FLN algorithm with 25 neurons achieved higher accuracy because of the knowledge rate is higher in FLN as mentioned in structure differences. And therefore, FLN also achieved better detection rate and less false alarm rate with less complexity of hidden neurons. Secondly, with all the improved of FLN that mentioned above, select randomly of the main FLN parameters considered as one of the algorithm limitations, which may not provide optimal parameters value and that represent a negative impact on the model accuracy. In general, random selection of

machine learning issues solved based proposed several optimization algorithms as mentioned in chapter 2. This work proposed PSO algorithm to be hybrid with basic FLN as training for FLN and reduces the impact of parameters random selected.

In table 4.2 represented results of a new model PSO-FLN, this section had been investigated the model accuracy based three parameters (number of particles (10-2-35-50), number of iterations (100-250-500), and number of neurons (10-25-35-50)). Accordingly, the influences of these changes based model accuracies were different. On another hand, the increase of a neuron's number was the most impact, results of compare the basic FLN and PSO-FLN models based same number of neurons shown in figures 4.6. Moreover, changes influence on detection rate and false alarm rate also, Where PSO-FLN with 50 neutrons and particles achieved better performance based these measurement's detection rate (DR), false alarm rate (FAR) with all iteration's numbers. In the following, figures are shown comparison results of basic FLN and PSO-FLN based DR and FAR measurements evaluated.

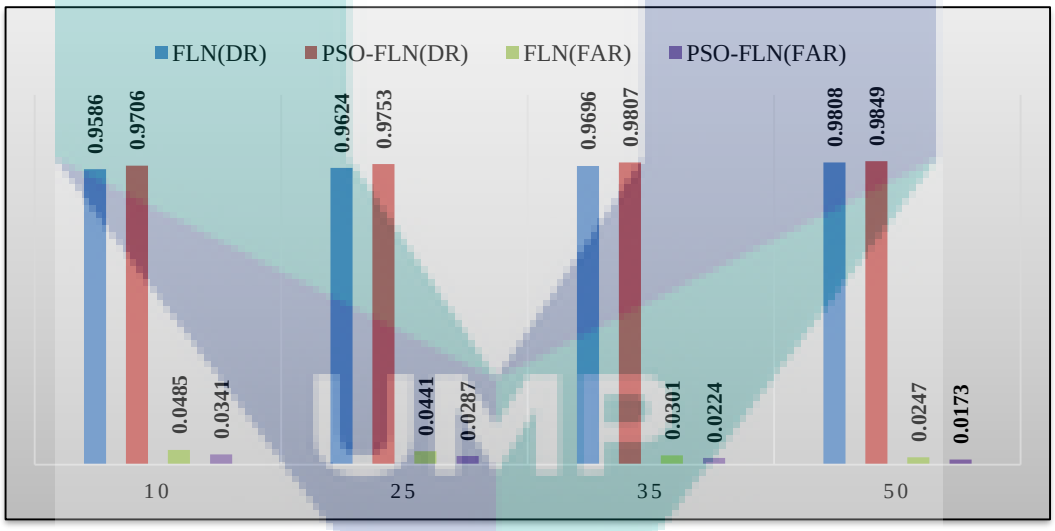


Figure 4.16 comparison result between FLN and PSO-FLN with 100 iterations

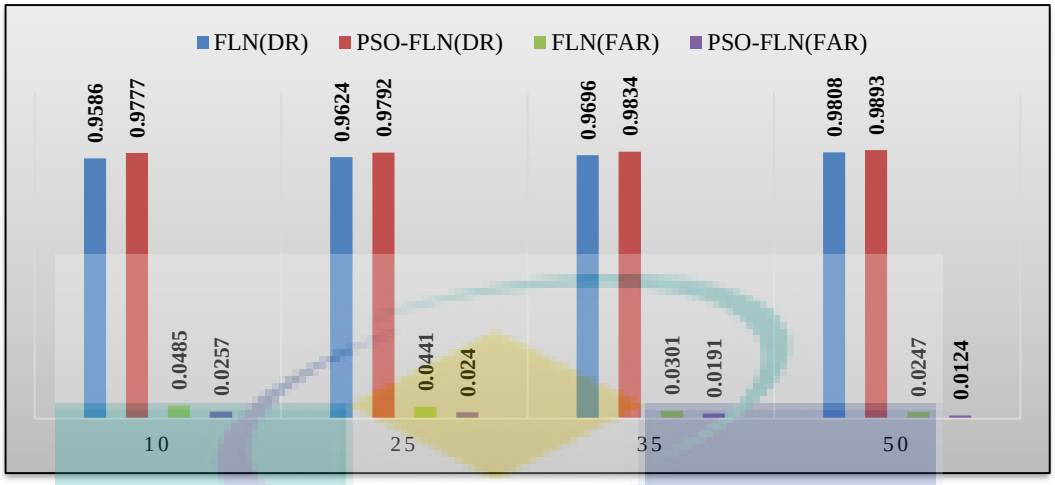


Figure 4.17comparison result between FLN and PSO-FLN with 250 iterations

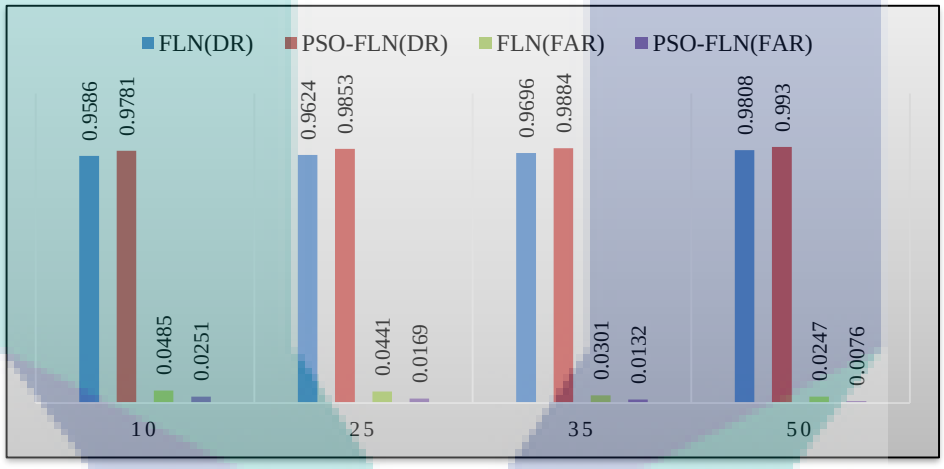


Figure 4.18 comparison result between FLN and PSO-FLN with 500 iterations

The figures above showed that the PSO-FLN performance improved the basic FLN performance by provided best parameter's values as shown in PSO-FLN block diagram figure 3.6. In the end, PSO-FLN improved showed how it is worth reducing the impact of select parameters randomly.

Thirdly, there are many works proposed to improved standard PSO algorithm as showed in previous chapters. The meeting room approach proposed in this work for parameters tuning by share parameter's information between several PSO-FLN models (clans). As showed in table 4.2, which it showed PSO-FLN best results with 50 particles and neurons with 500 iterations. However, proposed MRPSO-FLN with 5 clans each one represented PSO-FLN with 10 neurons and particles and 100 iterations.

Where in table 4.3 showed the comparison results between the best set of PSO-FLN (50 neurons and particles with 500 iterations) and MRPSO-FLN (10 neurons and particles with 100 iterations) and figures 4.7,4.8 and 4.9 shown how the new proposed MRSPO-FLN with less complexity achieved better performance.

Finally, the validation part is represented in the end of this chapter, which its divided into two parts, Compensation of PSO-FLN parameter's value (c1, c2, w) that provided from MRPSO-FLN instead of default values. The new model (PSO-FLN*) compared with PSO-FLN based (10 particles, 100 iterations) and (50 particles, 500 iterations) with adjust parameter values as default values as showed in table 4.4 and 4.5.

First table shows the results of comparison between PSO-FLN and PSO-FLN* with the same adjustment's situation. As a result of best parameter values provided for PSO-FLN* the performance such as accuracy 4.10, detection rate 4.11 and false alarm rate 4.12 improved in compared with PSO-FLN based default parameter's values. The new model PSO-FLN* achieved with only 10 neurons accuracy 0.9931 when PSO-FLN 0.9932 with 50 neurons, which mean the new proposed achieve comparable accuracy with less complexity. In addition, the second able shown the result comparison of PSO-FLN* with fewer numbers of particles and iterations achieved slightly better average accuracy as figure 4.13, and detection rate as shown in figure 4.14 and less false alarm rate as shown in figure 4.15.

Last part of validate, provide FLN parameters value taken from PSO-FLN* model instead of created randomly. The evaluated of a new model improved FLN (FLN) by compared with several standard models as shown in table 4.6. Subsequently, table 4.7 showed the results of comparison. IFLN model only with 10 neurons, showed improved as accuracy in figure 4.16, false alarm in figure 4.17 and detection rate in figure 4.18.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Overview

This chapter summarizes the thesis study and the achieved contributions. In addition, the chapter also defines the challenges and the future directions of research in order to enable fully machine learning based IDS. Section 5.2 highlights the objectives revisited. Section 5.3 presents brief summary of the research. Section 5.4 shows the recommendation for future research and limitation. Finally, section 5.5 gives the conclusion.

5.2 Objectives Revisited

This research was aimed at enhancing the accuracy of network intrusion detection system by using more recent artificial neural version called Fast Learning Network with improved particle swarm optimization based on a new multi-swarm scheme called Multi-Swarm Optimization (MRPSO). The objectives of this research are as follows:

i- To propose a self-parameters tuning technique for the Particle Swarm Optimization (PSO) algorithm using a multi-swarm approach (MRPSO).

To address the first objective, a new multi-swarm scheme inspired by the human social behaviour called MRPSO was developed and used to find the best values of the PSO control parameters as these parameters have an impact on the performance of the PSO algorithm. The proposed strategy was designed to interact with several PSO groups while searching for the optimal values of the control parameters. A new

cooperative of multi-swarm divided each group as clan and leaders based on the fitness value provided by the FLN. In each generation, the leaders often meet to select an overall best leader who will update the parameter values of the other leaders based on his new-found value

ii- To design a new training algorithm for FLN based on the proposed MRPSO algorithm for network intrusion detection system IDS.

As earlier stated, FLN is straightforward in implementation, computationally efficient, and have excellent learning performance characteristics. However, the randomness of the selected values of the main parameters (weight, basis) may not provide the optimal values which impact the accuracy of intrusion detection. This work proposed the MRPSO algorithm for training the FLN to reduce the impact of randomizes selection through the provision of the best values for these parameters. The fitness function was used as a rule to compare the generations of MRPSO and this fitness represents the level of accuracy of the FLN in providing false alarm rates (higher accuracy implies less false alarm rates). In the end, the mixed model provided better intrusion detection results compared to several other standard algorithms based on the NSL-KDD data set.

iii- To evaluate and test the prediction accuracy of proposed models (FLN, PSO-FLN, MRPSO-FLN) based on NSL-KDD dataset.

Based on the related works, this work proposed several standard metrics which can be utilized to evaluate and compare results derived from the different classification methods. The models were run for fifteen times and the average values for these runs were taken.

5.3 Brief Summary of Research

This study was motivated by the shortcoming of the prior approaches to network intrusion detection problems, such as high false alarm rates and poor detection performances. Three new machine learning models were introduced to network intrusion detection. The FLN, PSO-FLN, and MRPSO-FLN models are suitable for processing large multicast network intrusion detection datasets such as the

NSL-KDD dataset. The performance of the final model was compared to several standard algorithms such as Basic FLN, ELM, SVM, Multilayer Perceptron, Naive Bayes, and Decision Trees. Previous approaches to network intrusion detection yielded systems that have good detection performance for some classes of attacks but poor performance for others. Prior research on multiple algorithm approaches often lack a systematic method for combining the decisions of multiple learners or requires complex parameter settings, pre-processing or profiteering of data and human intervention.

In several literature instances, the approaches showed good detection performances but with a high rate of false positives which is a huge challenge for network operators. Other methods have performed well in detecting normal traffic and attack traffic but did not mention the attack classes that could potentially be involved in installing malware or destructive executable code. Similarly, several well-performing machine learning approaches are not readily scalable to handle larger datasets or multiclass problems such as those encountered in network intrusion detection.

The models introduced here addressed many of these issues as this study has demonstrated that both the PSO-FLN and FLN approaches performed well on the NSL-KDD dataset based on standard performance metrics compared to several standard algorithms and models. The MRPSO based on FLN with sigmoid function achieved the best overall performance among the various classifiers tested, with good detection performance, low misclassification rates, and very low false alarm rates.

The choices made for metrics and the way the model presents with different numbers of neurons in the hidden layer and number of swarm particles were considered to investigate the influence of different structures based on the accuracy of the models. Finally, the discovery that the multi-swarm based "meeting room" approach with FLN algorithm performed very well with the analyzed network traffic based on the NSL-KDD dataset should encourage more studies to include these parameters in the future studies and explore the influence of other activation functions on the performance of the models. As mentioned in the previous chapters, this research is the first to successfully adapt basic FLN algorithm and its hybrid variants (PSO-FLN and MRPSO-FLN algorithm) to the problem of network intrusion detection

5.4 Recommendation for Future Research and Limitation

There are many avenues for future research that could be explored, including a modern method for parameters selection, new classification combination methods, the addition of learning and experimentation with data pre-processing, feature selection, incremental and adversarial learning. Additionally, further research on the on-line processing techniques, hybrid approaches using partial batching with online techniques and unsupervised learning could be useful to produce practical intrusion-detection systems that could be utilized in a real-world network operating environment. Future research should also include testing the PSO-FLN and MRPSO-FLN methods with other intrusion-detection data sets. It could also be necessary to explore the use of newer approaches in combination with the PSO algorithm to improve upon the premature convergence problem and search performance of PSO when performing multimodal functions as discussed in chapter 2.

This research focused on reducing the rate of false positive detection while striving to get reasonable detection due to the needs of today's information technology professionals and the current environment of invasive cyber-attacks. As the network providers gain knowledge and develop more advanced approaches, attackers are adapting their tactics and changing their behaviours to thwart defences. Studies on the ways to combat these threats are important, while the development of systems that can adapt to the dynamism of the attackers through contextual and semantic learning, experimentation with data pre-processing, and intelligent feature selection holds promise. Future modifications on the PSO-FLN and MRPSO-FLN approaches could include temporal, sequential and context-based features with unsupervised learning to allow the models to readily detect anomalies. Moreover, this study encountered some limitation that can be considered in the future developments. The limitations of the study were as follows:

i- It has not been evaluated from the perspective of time execution. Considering the execution time of attack detection is useful for providing an estimation for the feasibility of operating such models in real-time.

ii- There was no incorporation of simulation models for cloud environments. Such incorporation is important for enabling extra features such as early attack detection and prevention.

iii- More attention can be paid to the unbalanced data aspect to improve the low accuracy of certain classes.

5.5 Conclusion

In this study, a systems engineering approach was followed for applying machine learning techniques to the problem of network intrusion detection. The relevant previous studies were reviewed and methods were explored to combine the classification decisions of a diverse set of "learners" in a manner that would produce consistently good results. Three new approaches to machine learning were introduced and tested against benchmark datasets, including the famous NSL-KDD intrusion detection dataset.

Three models were deployed in this work; firstly, the basic FLN was used to work as an intrusion detection system. The results of the FLN were compared to that of ELM algorithm to evaluate the proposed model. The results showed FLN to have a higher accuracy compared to ELM because of the FLN structure which comprised of a parallel connection of a multilayer feedforward neural network and a single-layer feedforward neural network. The DPFNN output nodes not only receive the recodification of the external information through the hidden nodes but also receives the external information itself directly through the input nodes. Secondly, to reduce the impact of randomized parameters select in machine learning algorithms, there are several optimizations algorithms and techniques that have been proposed and the FLN is one of the current approaches in the field of machine learning. FLN is also faced with the same limitation of random parameter selection. This work proposed the particle swarm optimization algorithm for training the FLN. The results of this model were compared to that of the basic FLN based on several standard evaluation criteria. Moreover, the performance of the new model (PSO-FLN) was better in terms of accuracy compared to that of the basic FLN. Thirdly, based on the literature evidence, most of the related works proposed PSO with default parameters values; on the other hand, there are several related works that proposed different methods for the

modification and improvement of the standard PSO in order to increase its ability to control the balance between exploration and exploitation or to enhance the search process of PSO.

Moreover, the standard PSO contains three control parameters and the selection of the values of these parameters depend on the standard or training and error. The values of these parameters have a great effect on the purposeful detecting capacity of the algorithm. This work proposed a new cooperative multi-swarm scheme called multi-swarm optimization (MRPSO) which was inspired by the human social behavior (the interaction between a group of people known as 'Clan' and their leaders). The proposed scheme consists of several swarms called clans and each clan consists of several solutions represented by the group members. The best member of each clan is the clan leader and has control over the members of its clan in terms of tuning the parameters. The selection of a leader from the clans is based on the fitness function which reflects the accuracy of FLN in comparing between different clans during the iterations. The clan with the best accuracy is selected as the leader. Furthermore, a new system called MRPSO-FLN was developed at the end of this process. The model was compared to the standard PSO-FLN with different numbers of neurons and iterations, and the output of the proposed algorithm showed a better accuracy or convergent results but with less number of neurons and number of iterations based on the NSL-KDD data set.

The last section of the study covered the validation of the proposed algorithm based on the values of the parameters achieved by the MRPSO-FLN instead of the default values of the parameters. Moreover, a new PSO-FLN with the best values of weight and bias was developed. This algorithm can be incorporated into basic FLN for parameter selection instead of a random parameter selection. Finally, the new algorithm was compared to several standards algorithms such as basic FLN, ELM, SVM, Naïve Bayes, Multilayer Perceptron, and Hoeffding tree (Decision tree) based on several standard evaluation measurements using the NSL-KDD data set.

REFERENCES

Aburomman, A. A., & Ibne Reaz, M. Bin. (2017). A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems. *Information Sciences*. https://doi.org/10.1016/j.ins.2017.06.007

Ahila, R., Sadasivam, V., & Manimala, K. (2015). An integrated PSO for parameter determination and feature selection of ELM and its application in classification of power system disturbances. *Applied Soft Computing Journal*. https://doi.org/10.1016/j.asoc.2015.03.036

Ahmad, A., & Senga, B. P. S. (2017). Instruction detection system based on support vector machine using bat algorithm. *International Journal of Computer Applications*. https://doi.org/10.5120/ijca2017912843

AL-Hawawreh, M., Moustafa, N., & Sitnikova, E. (2018). Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of Information Security and Applications*.https://doi.org/10.1016/j.jisa.2018.05.002

Al-Yaseen, W. L., Othman, Z. A., & Nazri, M. Z. A. (2017). Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2016.09.041

Ali, M., Khan, S. U., & Vasilakos (2015). Security in cloud computing : Opportunities and challenges. *Information Sciences*. https://doi.org/10.1016/j.ins.2015.01.025

Ambusaidi, M., He, X., Nanda, P., & Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Transactions on Computers*. https://doi.org/10.1109/TC.2016.2519914

Aslahi-Shahri, B. M., Rahmani, R., Chizari, M., Maralani, A., Eslami, M., Golkar, M. J., & Ebrahimi, A. (2016). A hybrid method consisting of GA and SVM for intrusion
detection system. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-015-1964-2

Baiad, R., Alhussein, O., Otrok, H., & Muhaidat, S. (2016). Novel cross layer detection schemes to detect blackhole attack against QoS-OLSR protocol in VANET. *Vehicular Communications*. https://doi.org/10.1016/j.vehcom.2016.09.001

Cao, J., Lin, Z., Huang, G. Bin, & Liu, N. (2012). Voting based extreme learning machine. *Information Sciences*. https://doi.org/10.1016/j.ins.2011.09.015

Cao, J., Zhao, Y., Lai, X., Ong, M. E. H., Yin, C., Koh, Z. X., & Liu, N. (2015). Landmark recognition with sparse representation classification and extreme learning machine. *Journal of The Franklin Institute*. https://doi.org/10.1016/j.jfranklin.2015.07.002

Chen, D., Chen, J., Jiang, H., Zou, F., & Liu, T. (2015). An improved PSO algorithm

based on particle exploration for function optimization and the modeling of chaotic systems. *Soft Computing*. https://doi.org/10.1007/s00500-014-1469-4

Chen, X. Y., & Chau, K. W. (2016). A hybrid double feedforward neural network for suspended sediment load estimation. *Water Resources Management*. https://doi.org/10.1007/s11269-016-1281-2

Choo, K. R. (2011). The cyber threat landscape: challenges and future research directions. *Computers & Security*. https://doi.org/10.1016/j.cose.2011.08.004

Deng, W. Y., Zheng, Q. H., Lian, S., Chen, L., & Wang, X. (2010). Ordinal extreme learning machine. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2010.08.022

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering*. https:// doi: 10.1109/tse.1987.232894

Diao, R., & Shen, Q. (2015). Nature inspired feature selection meta-heuristics. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-015-9428-8

Eberhart, R. C., & Yuhui Shi. (2001). Tracking and optimizing dynamic systems with particle swarms. *In the 2001 Congress on Evolutionary Computation*. https://doi.org/10.1109/CEC.2001.934376

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *In the 6th International Symposium on Micro Machine and Human Science*. https://doi.org/10.1109/MHS.1995.494215

Elhag, S., Fernández, A., Bawakid, A., Alshomrani, S., & Herrera, F. (2015). On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on Intrusion Detection Systems. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2014.08.002

Enache, A.-C., & Sgarciu, V. (2014). Anomaly intrusions detection based on support vector machines with bat algorithm. *In 18th International Conference on System Theory, Control and Computing*. https://doi.org/10.1109/ICSTCC.2014.6982526

Feng, G., Huang, G., Lin, Q., & Gay, R. (2009). Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*. https://10.1109/TNN.2009.2024147

Fossaceca, J. M., Mazzuchi, T. A., & Sarkani, S. (2015). MARK-ELM: Application of a novel multiple kernel learning framework for improving the robustness of network
intrusiondetection. *Expert Systems with Applications*.https://doi.org/10.1016/j.eswa.2014.12.040

Gauthama Raman, M. R., Somu, N., Kirthivasan, K., Liscano, R., & Shankar Sriram, V. S. (2017). An efficient intrusion detection system based on hypergraph - Genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2017.07.005

Ghaedi, M., Shojaeipour, E., Ghaedi, A. M., & Sahraei, R. (2015). Isotherm and

kinetics study of malachite green adsorption onto copper nanowires loaded on activated carbon: Artificial neural network modeling and genetic algorithm optimization. *Spectrochimica Acta: Molecular and Biomolecular Spectroscopy*. https://doi.org/10.1016/j.saa.2015.01.086

Goodarzi, B. G., Jazayeri, H., & Fateri, S. (2014). Intrusion detection system in computer network using hybrid algorithms (SVM and ABC). *Journal of Advances in Computer Research*. https://doi.org/10.1109/TC.2016.2519914

Gülcü, Ş., & Kodaz, H. (2015). A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization. *Engineering Applications of Artificial Intelligence*. https://doi.org/10.1016/j.engappai.2015.06.013

Hajimirzaei, B., & Navimipour, N. J. (2018). Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. *ICT Express*. https://doi.org/https://doi.org/10.1016/j.icte.2018.01.014

Hajisalem, V., & Babaie, S. (2018). A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Computer Networks*. https://doi.org/10.1016/j.comnet.2018.02.028

Han, F., & Huang, D. (2006). Improved extreme learning machine for function approximation by encoding a priori information. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2006.02.013

He, S., Wu, Q. H., Wen, J. Y., Saunders, J. R., & Paton, R. C. (2004). A particle swarm optimizer with passive congregation. *BioSystems* .https://doi.org/10.1016/j.biosystems.2004.08.003

Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P.-L., Iorkyase, E., Tachtatzis, C., & Atkinson, R. (2016). Threat analysis of IoT networks using artificial neural network intrusion detection system. *In International Symposium on Networks, Computers and Communications*. https://doi.org/10.1109/ISNCC.2016.7746067

Hosseini Bamakan, S. M., Wang, H., Yingjie, T., & Shi, Y. (2016). An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos
particle swarm optimization. *Neurocomputing* .https://doi.org/10.1016/j.neucom.2016.03.031

Huang, C. L., & Dun, J. F. (2008). A distributed PSO-SVM hybrid system with feature selection and parameter optimization. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2007.10.007

Huang, G.-B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems*. https://doi.org/10.1109/TSMCB.2011.2168604

Huang, G.-B., Zhu, Q.-Y., Qin, A. K., Suganthan, P. N., & Huang, G.-B. (2005). Evolutionary extreme learning machine. *Pattern Recognition*.https://doi.org/http://dx.doi.org/10.1016/j.patcog.2005.03.028

Huang, G.-B., Zhu, Q., Siew, C., Ã, G. H., Zhu, Q., (2006). Extreme learning machine: Theory and applications. *Neurocomputing* https://doi.org/10.1016/j.neucom.200 .12.126

Huang, G., Zhu, Q., & Siew, C. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. In *International Joint Conference on Neural Networks*. https://doi.org/10.1109/IJCNN.2004.1380068

Huang, H. X., Li, J. C., & Xiao, C. L. (2015). A proposed iteration optimization approach integrating backpropagation neural network with genetic algorithm. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.204.07.039

Huang, Y., Tang, J., Cheng, Y., Li, H., Campbell, K. A., & Han, Z. (2016). Real-time detection of false data injection in smart grid networks: An adaptive CUSUM method and analysis.*IEEE Systems Journal*.https://doi.org/10.1109/JSYST.2014 .2323266

Huang, G. Bin, Chen, L., & Siew, C. K. (2006). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*. https://doi.org/10.1109/TNN.2006.875977

Hubballi, N., & Suryanarayanan, V. (2014). False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications*. https://doi.org/10.1016/j.comcom.2014.04.012

Huynh, H. T., & Won, Y. (2008). Small number of hidden units for ELM with two-stage
linear model. *Transactions on Information and Systems*. https://doi.org/10.109 3/ietisy/e91-d.4.1042

Jahan, A., Mustapha, F., Ismail, Y., Sapuan, S. M., & Bahraminasab, M. (2011). A comprehensive VIKOR method for material selection A comprehensive VIKOR method for material selection. *Materials and Design*.4https://doi.org/10.1016/j. matdes.2010.10.015

Ji, S. Y., Jeong, B. K., Choi, S., & Jeong, D. H. (2016). A multi-level intrusion detection method for abnormal network behaviors. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2015.12.004

Jun-jie, X. U. (2005). An extended particle swarm optimizer. In *19th IEEE International*
Parallel and Distributed Processing Symposium.https://doi.org/10.1109/IPDPS. 2005.101

Kang, S.-H., & Kim, K. J. (2016). A feature selection approach to find optimal feature subsets for the network intrusion detection system. *Cluster Computing*. https://doi.org/10.1007/s10586-015-0527-8

Kaur, T., Malhotra, V., & Singh, D. (2014). Comparison of network security tools-Firewall, Intrusion Detection System and Honeypot. *International Journal of Enhanced Research in Science Technology & Engineering*,. https://doi.org/10.1109/JSYST.2014.2323266

Kavitha, K. M. C. A.& Phil, M. (2013). Particle swarm optimization for adaptive anomaly-based intrusion detection system using fuzzy controller. *International Journal of Network Security*. https://doi.org/10.1109/TSMCB.2011.2168604

Kevric, J., Jukic, S., & Subasi, A. (2017). An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-016-2418-1

Khan, Javed Akhtar, and N. J. (2016). A survey on intrusion detection systems and classification techniques. *International Journal of Scientific Research in Science, Engineering and Technology*.https://doi.org/http://dx.doi.org/10.1016/j.patcog.2005.03.028

Kiaee, F., Sheikhzadeh, H., & Eftekhari Mahabadi, S. (2015). Sparse Bayesian mixed-effects extreme learning machine, an approach for unobserved clustered heterogeneity. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2015.10.073

Kiranyaz, S., Ince, T., Yildirim, A., & Gabbouj, M. (2009). Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. *Neural Networks*. https://doi.org/10.1016/j.neunet.2009.05.013

Koc, L., Mazzuchi, T. A., & Sarkani, S. (2012). A network intrusion detection system based on a hidden naïve bayes multiclass classifier. *Expert Systems With Applications*. https://doi.org/10.1016/j.eswa.2012.07.009

Kumar, U. (2015). A survey on intrusion detection systems for cloud computing environment. *International Journal of Computer Applications.* https://doi.org/10.5120/19150-0573

Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2017). A survey of deep learning-based network anomaly detection. *Cluster Computing*. https://doi.org/10.1007/s10586-017-1117-8

Lan, Y., Soh, Y. C., & Huang, G. Bin. (2010). Two-stage extreme learning machine for regression. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2010.07.012

Latif, R., Abbas, H., Assar, S., & Ali, Q. (2014). Cloud computing risk assessment : a systematic literature review. *Future Information Technology*. https://doi.org/10.1007/978-3-642-40861-8

Li, G., Niu, P., Duan, X., & Zhang, X. (2014). Fast learning network: A novel artificial neural network with a fast learning speed. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-013-1398-7

Li, G., Niu, P., Wang, H., & Liu, Y. (2014). Least square fast learning network for modeling the combustion efficiency of a 300wm coal-fired boiler. *Neural Networks*. https://doi.org/10.1016/j.neunet.2013.12.006

Li, G., Qi, X., Chen, B., Ma, Y., Niu, P., & Chen, Z. (2017). Fast learning network with parallel layer perceptrons. *Neural Processing Letters*.https://doi.org/10.1007/s11063-017-9667-6

Li, X., Niu, P., Li, G., & Liu, J. (2017). An adaptive extreme learning machine for modeling nox emission of a 300 mw circulating fluidized bed boiler. *Neural Processing Letters*. https://doi.org/10.1007/s11063-017-9611-9

Li, M. Bin, Huang, G. Bin, Saratchandran, P., & Sundararajan, N. (2005). Fully complex
extreme learning machine. *Neurocomputing*.https://doi.org/10.1016/j.neucom.2005.03.002

Liang, G., Weller, S. R., Zhao, J., Luo, F., & Dong, Z. Y. (2017). The 2015 ukraine blackout: implications for false data injection attacks. *IEEE Transactions on Power Systems*. https://doi.org/10.1109/TPWRS.2016.2631891

Liang, N.-Y., Huang, G.-B., Saratchandran, P., & Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*. https://doi.org/10.1109/TNN.2006.880583

Liao, H. J., Richard Lin, C. H., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2012.09.004

Lin, W. C., Ke, S. W., & Tsai, C. F. (2015). CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2015.01.009

Liu, X., Li, P., & Gao, C. (2013). Symmetric extreme learning machine. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-012-0859-8

Lu, H., Du, B., Liu, J., Xia, H., & Yeap, W. K. (2017). A kernel extreme learning machine algorithm based on improved particle swam optimization. *Memetic Computing*. https://doi.org/10.1007/s12293-016-0182-5

Ludwig, S. A. (2017). Intrusion detection of multiple attack classes using a deep neural net ensemble. *In IEEE International Conference on Emerging Technologies and Factory Automation*. https://doi.org/10.1109/SSCI.2017.8280825

Ma, Y., Niu, P., Yan, S., & Li, G. (2018). A modified online sequential extreme learning machine for building circulation fluidized bed boiler's NOx emission model. *Applied Mathematics and Computation*. https://doi.org/10.1016/j.amc.2018.03.010

May, R. J., Maier, H. R., & Dandy, G. C. (2010). Data splitting for artificial neural networks using SOM-based stratified sampling. *Neural Networks* .https://doi.org/10.1016/j.neunet.2009.11.009

Mirjalili, S., Zaiton, S., Hashim, M., & Sardroudi, H. M. (2012). Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*. https://doi.org/10.1016/j.amc.2012.04.069

Mishra, P., Pilli, E. S., Varadharajan, V., & Tupakula, U. (2016). Intrusion detection techniques in cloud environment: A Survey. *Journal of Network and Computer*

Applications. https://doi.org/10.1016/j.jnca.2016.10.015

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., & Rajarajan, M. (2013a). A survey of intrusion detection techniques in Cloud. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2012.05.003

Moradi, M., & Zulkernine, M. (2004). A neural network based system for intrusion detection and classification of attacks. *In IEEE International Conference on Advances in Intelligent Systems-Theory and Applications*. https://doi.org/10.1016/j.amc.2012.04.069

Moustafa, N., Slay, J., & Creech, G. (2017). Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. *IEEE Transactions on Big Data*. https://doi.org/10.1109/TBDATA.2017.2715166

Mukherjee, S., & Sharma, N. (2012). Intrusion detection using naive bayes classifier with
featurereduction. *Procedia Technology*. https://doi.org/10.1016/j.protcy.2012.05.017

Niu, B., Zhu, Y., He, X., & Wu, H. (2007). MCPSO: A multi-swarm cooperative particle
swarmoptimizer. *Applied Mathematics and Computation*. https://doi.org/10.1016/j.amc.2006.07.026

Niu, P., Chen, K., Ma, Y., Li, X., Liu, A., & Li, G. (2017). Model turbine heat rate by fast learning network with tuning based on ameliorated krill herd algorithm. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2016.11.011

Niu, P., Ma, Y., Li, M., Yan, S., & Li, G. (2016). A kind of parameters self-adjusting extreme learning machine. *Neural Processing Letters*. https://doi.org/10.1007/s11063-016-9496-z

Patel, A., Taghavi, M., Bakhtiyari, K., & Celestino Júnior, J. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2012.08.007

Pervez, M. S., & Farid, D. M. (2014). Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. *In 8th International Conference on Software, Knoledgw, Information Management and Applications(SKIMA 2018)*. https://doi.org/10.1109/SKIMA.2014.7083539

Phoungphol, P., Zhang, Y., & Zhao, Y. (2012). Robust multiclass classification for learning from imbalanced biomedical data. *Tsinghua Science and Technology*. https://doi.org/10.1109/TST.2012.6374363

Pornsing, C., Sodhi, M. S., & Lamond, B. F. (2016). Novel self-adaptive particle swarm optimization methods. *Soft Computing*. https://doi.org/10.1007/s00500-015-1716-3

Raghav, I. (2013). Intrusion detection and prevention in cloud environment : A Systematic Review. *International Journal of Computer Applications*. https://doi.org/10.1109/TPAMI.2009.485

Raja, M. A. Z. (2014). Stochastic numerical treatment for solving Troesch's problem. *Information Sciences*. https://doi.org/10.1016/j.ins.2014.04.036

Raja, M. A. Z., Shah, F. H., Tariq, M., Ahmad, I., & Ahmad, S. ul I. (2016). Design of artificial neural network models optimized with sequential quadratic programming to study the dynamics of nonlinear Troesch's problem arising in plasma physics. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-016-2530-2

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2004.826071

Rodríguez, J. D., Pérez, A., & Lozano, J. A. (2010). Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2009.187

Rong, C., Nguyen, S. T., & Gilje, M. (2012). A survey on security challenges in cloud computing. *Computer and Electrical Engineering*. https://doi.org/10.1016/j.compeleceng.2012.04.015

Rong, H. J., Ong, Y. S., Tan, A. H., & Zhu, Z. (2008). A fast pruned-extreme learning machine for classification problem. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2008.01.005

Huang, R., & He, M. (2007). Feature selection using double parallel feedforward neural networks and particle swarm optimization. *In 2007 IEEE Congress on Evolutionary Computation*. https://doi.org/10.1016/j.ipl.2004.11.003

Sabhnani, M., & Serpen, G. (2004). Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set. *Intelligent Data Analysis*. https://doi.org/10.1007/978-3-540-88623-5_41

Saxena MTech Scholar, H., & Richaariya, V. (2014). Intrusion detection in KDD99 dataset using SVM-PSO and feature reduction with information Gain. *International Journal of Computer Applications*. https://doi.org/11.1016/j.neucom.2007.01.018

Scardapane, S., & Wang, D. (2017). Randomness in neural networks: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. https://doi.org/10.1002/widm.1200

Scarfone, K., Mell, P., & Mell, P. (2012). Guide to intrusion detection and prevention systems (IDPS) recommendations of the national institute of standards and technology. *National Institute of Standards and Technology*. https://doi.org/2012arXiv1206.2944

Shah, S. A. R., & Issac, B. (2018). Performance comparison of intrusion detection systems and application of machine learning to Snort system. *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2017.10.016

Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C., & Wang, L. M. (2005). An improved GA and a novel PSO-GA-based hybrid algorithm. *Information Processing Letters*. https://doi.org/10.1016/j.ipl.2004.11.003

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *In IEEE International Conference on Evolutionary Computation*. https://doi.org/10.1109/ICEC.1998.699146

Shi, Y., & Eberhart, R. (1999). Empirical study of particle swarm optimization. *The Evolutionary Computation*. https://doi.org/10.1109/CEC.1999.785511

Sinan Q. Salih1,2 , AbdulRahman A. Alsewari1, Bellal Al-Khateeb2, M. F. Z. (2018). Novel multi-swarm approach for balancing exploration and exploitation in particle swarm optimization. *In International Conference of Reliable Information and Communication Technology* .https://doi.org/10.1109/NAFOSTED.2017.8108

Singh, R., Kumar, H., & Singla, R. K. (2015). An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2015.07.015

Sivatha, S. S., Geetha, S., & Kannan, A. (2012). Decision tree based light weight intrusion detection using a wrapper approach. *Expert Systems With Applications*. https://doi.org/10.1016/j.eswa.2011.06.013

Slowik, A., & Kwasnicka, H. (2018). Nature inspired methods and their industry applications-swarm intelligence algorithms. *IEEE Transactions on Industrial Informatics*. https://doi.org/10.1109/TII.2017.2786782

Snoek, J., Larochelle, H., & Adams, R. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*. https://doi.org/2012arXiv1206.2944S

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification4tasks. *Information Processing and Management*. https://doi.org/10.1016/j.ipm.2009.03.002

Sun, Y., Kamel, M. S., Wong, A. K. C., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*. https://doi.org/10.1016/j.patcog.2007.04.009

Tanweer, M. R., Suresh, S., & Sundararajan, N. (2015). Self regulating particle swarm optimization algorithm. *Information Sciences*. https://doi.org/10.1016/j.ins.2014.09.053

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *In IEEE Symposium on Computational Intelligence for Security and Defense Applications*.https://doi.org/10.1109/CISDA.2009.5356528

Tran, C., Vo, T. N., & Thinh, T. N. (2017). HA-IDS: A heterogeneous anomaly-based intrusion detection system. *In 4th Nafosted Conference on Information and Computer Science*. https://doi.org/10.1109/NAFOSTED.2017.8108056

Tsai, C., Hsu, Y., Lin, C., & Lin, W. (2009). Intrusion detection by machine learning : A review. *Expert Systems With Applications*. https://doi.org/10.1016/j.eswa.2009.05.029

Udaya Sampath K. Perera Miriya Thanthrige, Jagath Samarabandu, X. W. (2016). Machine learning techniques for intrusion detection. *In IEEE Canadian Conference on Electrical and Computer Engineering*.https://doi.org/10.1145/2980258.2980378

Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M., & Fischer, M. (2015). Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys*. https://doi.org/10.1016/j.asoc.2018.02.042

Von Solms, R., & Van Niekerk, J. (2013). From information security to cyber security. *Computers and Security*. https://doi.org/10.1016/j.cose.2013.04.004

Wang, C. R., Xu, R. F., Lee, S. J., & Lee, C. H. (2018). Network intrusion detection using equality constrained-optimization-based extreme learning machines. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2018.02.015

Wang, N., Er, M. J., & Han, M. (2014). Parsimonious extreme learning machine using recursive orthogonal least squares. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2013.2296048

Whitman, M. E., & Mattord, H. J. (2012). Principles of information security. *Course Technology*. https://doi.org/10.1016/B978-0-12-381972-7.00002-6

Xia, X., Gui, L., & Zhan, Z. H. (2018). A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2018.02.042

Xiang, J., Westerlund, M., Sovilj, D., & Pulkkis, G. (2014). Using extreme learning machine for intrusion detection in a big data environment. *In the 2014 Workshop on Artificial Intelligentand Security*.https://doi.org/10.1145/2666652.2666664

Zeng, N., Zhang, H., Liu, W., Liang, J., & Alsaadi, F. E. (2017). A switching delayed PSO optimized extreme learning machine for short-term load forecasting. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2017.01.090

Zhan, Z.-H., Zhang, J., Li, Y., & Chung, H. S.-H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems and Cybernetics Society*. https://doi.org/10.1109/TSMCB.2009.2015956

Zhang, L., Tang, Y., Hua, C., & Guan, X. (2015). A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2014.11.018

Zhang, L., & Zhang, D. (2017). Evolutionary Cost-Sensitive Extreme Learning Machine.
IEEE Transactions on Neural Networks and Learning Systems. https://doi.org/10.1109/TNNLS.2016.2607757

Aburomman, A. A., & Ibne Reaz, M. Bin. (2017). A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems. *Information Sciences*. https://doi.org/10.1016/j.ins.2017.06.007

Ahila, R., Sadasivam, V., & Manimala, K. (2015). An integrated PSO for parameter determination and feature selection of ELM and its application in classification of power system disturbances. *Applied Soft Computing Journal*. https://doi.org/10.1016/j.asoc.2015.03.036

Ahmad, A., & Senga, B. P. S. (2017). Instruction detection system based on support vector machine using bat algorithm. *International Journal of Computer Applications*. https://doi.org/10.5120/ijca2017912843

AL-Hawawreh, M., Moustafa, N., & Sitnikova, E. (2018). Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of Information Security and Applications*. https://doi.org/10.1016/j.jisa.2018.05.002

Al-Yaseen, W. L., Othman, Z. A., & Nazri, M. Z. A. (2017). Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2016.09.041

Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). Security in cloud computing: opportunities and challenges. *Information Sciences*. https://doi.org/10.1016/j.ins.2015.01.025

Ambusaidi, M., He, X., Nanda, P., & Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Transactions on Computers*. https://doi.org/10.1109/TC.2016.2519914

Approximation, F., & Problems, C. (2009). Online sequential fuzzy extreme learning machine for function approximation and classification problem. *IEEE Transactions on Systems*. https://10.1109/TSMCB.2008.2010506

Aslahi-Shahri, B. M., Rahmani, R., Chizari, M., Maralani, A., Eslami, M., Golkar, M. J., & Ebrahimi, A. (2016). A hybrid method consisting of GA and SVM for intrusion detection system. *Neural Computing and Applications*.https://doi.org/10.1007/s00521-015-1964-2

Baiad, R., Alhussein, O., Otrok, H., & Muhaidat, S. (2016). Novel cross layer detection schemes to detect blackhole attack against QoS-OLSR protocol in VANET. *Vehicular Communications*. https://doi.org/10.1016/j.vehcom.

Cao, J., Lin, Z., Huang, G. Bin, & Liu, N. (2012). Voting based extreme learning machine. *Information Sciences*. https://doi.org/10.1016/j.ins.

Cao, J., Zhao, Y., Lai, X., Ong, M. E. H., Yin, C., Koh, Z. X., & Liu, N. (2015). Landmark recognition with sparse representation classification and extreme

learning machine. *Journal of the Franklin Institute*. https://doi.org/10.1016/j.jfranklin.

Chen, D., Chen, J., Jiang, H., Zou, F., & Liu, T. (2015). An improved PSO algorithm based on particle exploration for function optimization and the modelling of chaotic systems. *Soft Computing*. https://doi.org/10.1007/s00500-014-1469-4

Chen, X. Y., & Chau, K. W. (2016). A hybrid double feedforward neural network for suspended sediment load estimation. *Water Resources Management*. https://doi.org/10.1007/s11269-016-1281-2

Choo, K. R. (2011). The cyber threat landscape: challenges and future research directions. *Computers & Security*. https://doi.org/10.1016/j.cose.2011.08.004

Deng, W. Y., Zheng, Q. H., Lian, S., Chen, L., & Wang, X. (2010). Ordinal extreme learning machine. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2010.08.022

Diao, R., & Shen, Q. (2015). Nature inspired feature selection meta-heuristics. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-015-9428-8

Eberhart, R. C. (2001). Fuzzy adaptive particle swarm optimization. *In Conference on Evolutionary Computation*. https://doi.org/10.1109/CEC.2001.934377

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *In the 6th International Symposium on Micro Machine and Human Science*. https://doi.org/10.1109/MHS.1995.494215

Elhag, S., Fernández, A., Bawakid, A., Alshomrani, S., & Herrera, F. (2015). On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2014.08.002

Enache, Sgarciu, V. (2014). Anomaly intrusions detection based on support vector machines with bat algorithm. *In 18th International Conference on System Theory*. https://doi.org/10.1109/ICSTCC.2014.6982526

Fossaceca, J. M., Mazzuchi, T. A., & Sarkani, S. (2015). MARK-ELM: Application of a novel multiple kernel learning framework for improving the robustness of network
intrusion detection. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2014.12.040

Gauthama Raman, M. R., Somu, N., Kirthivasan, K., Liscano, R., & Shankar Sriram, V. S. (2017). An efficient intrusion detection system based on hypergraph - Genetic algorithm for parameter optimization and feature selection in support vector machine. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2017.07.005

Ghaedi, M., Shojaeipour, E., Ghaedi, A. M., & Sahraei, R. (2015). Isotherm and kinetics study of malachite green adsorption onto copper nanowires loaded on

activated carbon: Artificial neural network modelling and genetic algorithm optimization.
Molecular and Biomolecular Spectroscopy.https://doi.org/10.1016/j.saa.2015.01.086

Gülcü, Ş., & Kodaz, H. (2015). A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization. *Engineering Applications of Artificial Intelligence*. https://doi.org/10.1016/j.engappai.2015.06.013

Hajimirzaei, B., & Navimipour, N. J. (2018). Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. *ICT Express*. https://doi.org/https://doi.org/10.1016/j.icte.2018.01.014

Hajisalem, V., & Babaie, S. (2018). A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Computer Networks*. https://doi.org/10.1016/j.comnet.2018.02.028

Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P.-L., Iorkyase, E., Tachtatzis, C., & Atkinson, R. (2016). Threat analysis of IoT networks using artificial neural network intrusion detection system. *In International Symposium on Networks, Computers and Communications*. https://doi.org/10.1109/ISNCC.2016.7746067

Hosseini Bamakan, S. M., Wang, H., Yingjie, T., & Shi, Y. (2016). An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos
particle swarm optimization. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2016.03.031

Huang, C. L., & Dun, J. F. (2008). A distributed PSO-SVM hybrid system with feature selection and parameter optimization. *Applied Soft Computing*, https://doi.org/10.1016/j.asoc.2007.10.007

Huang, G.-B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems*. https://doi.org/10.1109/TSMCB.2011.2168604

Huang, G.-B., Zhu, Q.-Y., Qin, A. K., Suganthan, P. N., & Huang, G.-B. (2005). Evolutionary extreme learning machine. *Pattern Recognition.* https://doi.org/http://dx.doi.org/10.1016/j.patcog.2005.03.028

Huang, G.-B., Zhu, Q., Siew, C., Ã, G. H., Zhu, Q., Siew. (2006). Extreme learning machine:Theory and applications. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2005.12.126

Huang, G., Zhu, Q., & Siew, C. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. *In International Conference on Neural Networks*. https://doi.org/10.1109/IJCNN.2004.1380068

Huang, H. X., Li, J. C., & Xiao, C. L. (2015). A proposed iteration optimization approach integrating backpropagation neural network with genetic algorithm. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2014.07.039

Huang, Y., Tang, J., Cheng, Y., Li, H., Campbell, K. A., & Han, Z. (2016). Real-time detection of false data injection in smart grid networks: An adaptive CUSUM method and analysis. *IEEE Systems Journal*. https://doi.org/10.1109/JSYST.2014.2323266

Huang, G. Bin, Chen, L., & Siew, C. K. (2006). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*. https://doi.org/10.1109/TNN.2006.875977

Hubballi, N., & Suryanarayanan, V. (2014). False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications*. https://doi.org/10.1016/j.comcom.2014.04.012

Jahan, A., Mustapha, F., Ismail, Y., Sapuan, S. M., & Bahraminasab, M. (2011). A comprehensive VIKOR method for material selection. *Materials and Design*. https://doi.org/10.1016/j.matdes.2010.10.015

Ji, S. Y., Jeong, B. K., Choi, S., & Jeong, D. H. (2016). A multi-level intrusion detection method for abnormal network behaviours. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2015.12.004

Jun-jie, X. U. (2005). An extended particle swarm optimizer. In *19th IEEE International Parallel and Distributed Processing Symposium*. https://doi.org/10.1109/IPDPS.2005.101

Kang, S.-H., & Kim, K. J. (2016). A feature selection approach to find optimal feature subsets for the network intrusion detection system. *Cluster Computing*. https://doi.org/10.1007/s10586-015-0527-8

Kavitha., & Phil, M. (2013). Particle swarm optimization for adaptive anomaly-based intrusion detection system using fuzzy controller. *International Journal of Network Security*. http://dx.doi.org/10.1016/j.patcog.2005.03.028

Kevric, J., Jukic, S., & Subasi, A. (2017). An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-016-2418-1

Kiaee, F., Sheikhzadeh, H., & Eftekhari Mahabadi, S. (2015). Sparse Bayesian mixed-effects extreme learning machine, an approach for unobserved clustered heterogeneity. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2015.10.073

Kiranyaz, S., Ince, T., Yildirim, A., & Gabbouj, M. (2009). Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. *Neural*

Networks. https://doi.org/10.1016/j.neunet.2009.05.013

Koc, L., Mazzuchi, & Sarkani, S. (2012). A network intrusion detection system based on
a Hidden Naïve Bayes multiclass classifier. *Expert Systems With Application s*.https://doi.org/10.1016/j.eswa.2012.07.009

Kumar, U. (2015). A survey on intrusion detection systems for cloud computing environment. *International Journal of Computer Applications*. https://doi.org/10.5120/19150-0573

Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., & Kim, K. J. (2017). A survey of deep learning-based network anomaly detection. *Cluster Computing*. https://doi.org/10.1007/s10586-017-1117-8

Lan, Y., Soh, Y. C., & Huang, G. Bin. (2010). Two-stage extreme learning machine for regression. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2010.07.012

Li, G., Niu, P., & Duan, X. (2013). Fast learning network: a novel artificial neural network with a fast learning speed. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-013-1398-7

Li, G., Niu, P., Wang, H., & Liu, Y. (2014). Least square fast learning network for modelling the combustion efficiency of a 300WM coal-fired boiler. *Neural Networks*. https://doi.org/10.1016/j.neunet.2013.12.006

Li, G., Qi, X., Chen, B., Ma, Y., Niu, P., & Chen, Z. (2017). Fast learning network with parallel layer perceptrons. *Neural Processing Letters*. https://doi.org/10.1007/s11063-017-9667-6

Li, X., Niu, P., Li, G., & Liu, J. (2017). An adaptive extreme learning machine for modeling nox emission of a 300 mw circulating fluidized bed boiler. *Neural Processing Letters*. https://doi.org/10.1007/s11063-017-9611-9

Li, M. Bin, Huang, G. Bin, Saratchandran, P., & Sundararajan, N. (2005). Fully complex
extreme learning machine. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2005.03.002

Liang, G., Weller, S. R., Zhao, J., Luo, F., & Dong, Z. Y. (2017). The 2015 Ukraine blackout: implications for false data injection attacks. *IEEE Transactions on Power Systems*. https://doi.org/10.1109/TPWRS.2016.2631891

Liang, N.-Y., Huang, G.-B., Saratchandran, P., & Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*. https://doi.org/10.1109/TNN.2006.880583

Liao, H. J., Richard Lin, C. H., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*.

https://doi.org/10.1016/j.jnca.2012.09.004

Lin, W. C., Ke, S. W., & Tsai, C. F. (2015). CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2015.01.009

Liu, X., Li, P., & Gao, C. (2013). Symmetric extreme learning machine. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-012-0859-8

Lu, H., Du, B., Liu, J., Xia, H., & Yeap, W. K. (2017). A kernel extreme learning machine algorithm based on improved particle swam optimization. *Memetic Computing*. https://doi.org/10.1007/s12293-016-0182-5

Ludwig, S. A. (2017). Intrusion detection of multiple attack classes using a deep neural net ensemble. *In IEEE International Conference on Emerging Technologies and Factory Automation.* https://doi.org/10.1109/SSCI.2017.8280825

Ma, Y., Niu, P., Yan, S., & Li, G. (2018). A modified online sequential extreme learning machine for building circulation fluidized bed boiler's NOx emission model. *Applied Mathematics and Computation*. https://doi.org/10.1016/j.amc.2018.03.010

May, R. J., Maier, H. R., & Dandy, G. C. (2010). Data splitting for artificial neural networks using SOM-based stratified sampling. *Neural Networks* https://doi.org/10.1016/j.neunet.2009.11.009

Mirjalili, S., Zaiton, S., Hashim, M., & Sardroudi, H. M. (2012). Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*. https://doi.org/10.1016/j.amc.2012.04.069

Mishra, P., Pilli, E. S., Varadharajan, V., & Tupakula, U. (2016). Intrusion detection techniques in cloud environment: A Survey. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2016.10.015

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., & Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2012.05.003

Moradi, M., & Zulkernine, M. (2004). A neural network based system for intrusion detection and classification of attacks. *In IEEE International Conference on Advances in Intelligent Systems-Theory and Applications*. https://doi.org/10.1016/j.neucom.2010.07.012

Moustafa, N., Slay, J., & Creech, G. (2017). Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. *IEEE Transactionson Big Data*.https://doi.org/10.1109/TBDATA.2017.2715166

Mukherjee, S., & Sharma, N. (2012). Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technology*. https://doi.org/10.1016/j.protcy.2012.05.017

Niu, B., Zhu, Y., He, X., & Wu, H. (2007). MCPSO: A multi-swarm cooperative particle
swarm optimizer. *Applied Mathematics and Computation*. https://doi.org/10.1016/j.amc.2006.07.026

Niu, P., Chen, K., Ma, Y., Li, X., Liu, A., & Li, G. (2017). Model turbine heat rate by fast learning network with tuning based on ameliorated krill herd algorithm. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.2016.11.011

Niu, P., Ma, Y., Li, M., Yan, S., & Li, G. (2016). A kind of parameters self-adjusting extreme learning machine. *Neural Processing Letters*. https://doi.org/10.1007/s11063-016-9496-z

Okulewicz, M., & Mandziuk, J. (2015). Two-phase multi-swarm PSO and the dynamic vehicle routing problem. *In IEEE Symposium Series on Computational Intelligence*. https://doi.org/10.1109/CIHLI.2014.7013391

Patel, A., Taghavi, M., Bakhtiyari, K., & Celestino Júnior, J. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2012.08.007

Pervez, M. S., & Farid, D. M. (2014). Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. *In 8th International Conference on Software, Knowledge, Information Management and Applications*. https://doi.org/10.1109/SKIMA.2014.7083539

Phoungphol, P., Zhang, Y., & Zhao, Y. (2012). Robust multiclass classification for learning from imbalanced biomedical data. *Tsinghua Science and Technology*. https://doi.org/10.1109/TST.2012.6374363

Pornsing, C., Sodhi, M. S., & Lamond, B. F. (2016). Novel self-adaptive particle swarm optimization methods. *Soft Computing*. https://doi.org/10.1007/s00500-015-1716-3

Raja, M. A. Z. (2014). Stochastic numerical treatment for solving Troesch's problem. *Information Sciences*. https://doi.org/10.1016/j.ins.2014.04.036

Raja, M. A. Z., Shah, F. H., Tariq, M., Ahmad, I., & Ahmad, S. ul I. (2016). Design of artificial neural network models optimized with sequential quadratic programming to study the dynamics of nonlinear Troesch's problem arising in plasma physics. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-016-2530-2

Ratnaweera, A., Halgamuge,, & Watson,(2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*. https://doi.org/10.1109/TEVC.2004.826071

Rodríguez, J. D., Pérez, A., & Lozano, J. A. (2010). Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2009.187

Rong, C., Nguyen, S. T., & Gilje, M. (2012). Beyond lightning: A survey on security challenges in cloud computing. *Computer and Electrical Engineering*. https://doi.org/10.1016/j.compeleceng.2012.04.015

Rong, H. J., Ong, Y. S., Tan, A. H., & Zhu, Z. (2008). A fast pruned-extreme learning machine for classification problem. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2008.01.005

Sabhnani, M., & Serpen, G. (2004). Why machine learning algorithms fail in misuse detection on KDD intrusion detection data set. *Intelligent Data Analysis*. https://doi.org/10.1007/978-3-540-88623-5_41

Saxena, H., & Richaariya, V. (2014). Intrusion detection in KDD99 dataset using SVM-PSO and feature reduction with information gain. *International Journal of Computer Applications*. https://doi.org/10.1109/CIHLI.012

Scardapane, S., & Wang, D. (2017). Randomness in neural networks: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. https://doi.org/10.1002/widm.1200

Shah, S. A. R., & Issac, B. (2018). Performance comparison of intrusion detection systems and application of machine learning to Snort system. *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2017.10.016

Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C., & Wang, L. M. (2005). An improved GA and a novel PSO-GA-based hybrid algorithm. *Information Processing Letters*. https://doi.org/10.1016/j.ipl.2004.11.003

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *In IEEE International Conference on Evolutionary Computation*.https://doi.org/10.1109/ICEC.1998.699146

Shi, Y., & Eberhart, R. (1999). Empirical study of particle swarm optimization. *Evolutionary Computation*. https://doi.org/10.1109/CEC.1999.785511

Singh, R., Kumar, H., & Singla, R. K. (2015). An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2015.07.015

Sivatha, S. S., Geetha, S., & Kannan, A. (2012). Decision tree based light weight intrusion detection using a wrapper approach. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2011.06.013

Slowik, A., & Kwasnicka, H. (2018). Nature inspired methods and their industry applications-swarm intelligence algorithms. *IEEE Transactions on Industrial*

Informatics. https://doi.org/10.1109/TII.2017.2786782

Snoek, J., Larochelle, H., & Adams, R. (2012). Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information*. https://doi.org/2012arXiv1206.2944S

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*. https://doi.org/10.1016/j.ipm.2009.03.002

Su., Kamel,, Wong,., & Wang,. (2007). Cost-sensitive boosting for classification of imbalance data. *Pattern Recognition*, https://doi.org/10.1016/j.patcog.2007.04.009

Tanweer, M. R., Suresh, S., & Sundararajan, N. (2015). Self-regulating particle swarm optimization algorithm. *Information Sciences*. https://doi.org/10.1016/j.ins.2014.09.053

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *In IEEE Symposium on Computational Intelligence for Security and Defense Applications*. https://doi.org/10.1109/CISDA.2009.5356528

Tran, C., Vo, T. N., & Thinh, T. N. (2017). A heterogeneous anomaly-based intrusion detection system. *In 4th Conference on Information and Computer Science*. https://doi.org/10.1109/NAFOSTED.2017.8108056

Tsai, C., Hsu, Y., Lin, C., & Lin, W. (2009).Intrusion detection by machine learning: A review. *Expert Systems with Application*. https://doi.org/10.1016/j.eswa.2009.05.029

Udaya Sampath K. Perera Miriya Thanthrige, Jagath Samarabandu (2016). Machine learning techniques for intrusion detection. *In IEEE Canadian Conference on Electrical and Computer Engineering(CCECE)*. https://doi.org/10.1145/2980258.2980378

Von Solms, R., & Van Niekerk, J. (2013). From information security to cyber security. *Computers and Security*. https://doi.org/10.1016/j.cose.04.004

Wang, C. R., Xu, R. F., Lee, S. J., & Lee, C. H. (2018). Network intrusion detection using equality constrained-optimization-based extreme learning machines. *Knowledge-Based Systems*. https://doi.org/10.1016/j.knosys.02.015

Wang, J., Wu, W., Li, Z., & Li, L. (2011). Convergence of gradient method for double parallel feedforward neural network. *International Journal of Numerical Analysis and Modeling*. https://doi.org/10.1109/TPWRS.2016.2631891

Wang, N., Er, M. J., & Han, M. (2014). Parsimonious extreme learning machine using recursive orthogonal least squares. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2013.2296048

Wills, C. (2010). A survey of intrusion detection and prevention systems. *Information Management & Computer Security*. https://doi.org/10.1108/09685221011079199

Xia, X., Gui, L., & Zhan, Z. H. (2018). A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2018.02.042

Xiang, J., Westerlund, M., Sovilj, D., & Pulkkis, G. (2014). Using extreme learning machine for intrusion detection in a big data environment. *In 2014 Workshop on Artificial Intelligent and Security Workshop*.https://doi.org/10.1145/2666652.2666664

Zeng, N., Zhang, H., Liu, W., Liang, J., & Alsaadi, F. E. (2017). A switching delayed PSO optimized extreme learning machine for short-term load forecasting. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2017.01.090

Zhan, Z.-H., Zhang, J., Li, Y., & Chung, H. S.-H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems*. https://doi.org/10.1109/TSMCB.2009.2015956

Zhang, L., Tang, Y., Hua, C., & Guan, X. (2015). A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2014.11.018

Zhang, L., & Zhang, D. (2017). Evolutionary cost-sensitive extreme learning machine. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2016.2607757

APPENDIX A
EXPERIMENTS RESULTS OF ELM VS FLN WITH SEVERAL RUNS

10 number of neurons		25 number of neurons	
ELM	**FLN**	**ELM**	**FLN**
Run 1 Accuracy is: 0.90242 Precision: 0.90242 Recall: 0.90242 FM:: 0.90242 G-Mean:: 0.81746 Detection Rate: 0.90128 False Alarm Rate: 0.11995	Run 1 Accuracy is: 0.95781 Precision: 0.95781 Recall: 0.95781 FM:: 0.95781 G-Mean:: 0.91807 Detection Rate: 0.96 False Alarm Rate: 0.046812	Run=1 Testing Accuracy is: 0.94873 Precision: 0.94873 Recall: 0.94873 FM:: 0.94873 G-Mean:: 0.90104 Detection Rate: 0.93925 False Alarm Rate: 0.073042	Run = 1 Testing Accuracy is: 0.96731 Precision: 0.96731 Recall: 0.96731 FM:: 0.96731 G-Mean:: 0.93605 Detection Rate: 0.96223 False Alarm Rate: 0.044355
Run =2 Accuracy is: 0.91574 Precision: 0.91574 Recall: 0.91574 FM:: 0.91574 G-Mean:: 0.84102 Detection Rate: 0.91703 False Alarm Rate: 0.1001	Run =2 Accuracy is: 0.95819 Precision: 0.95819 Recall: 0.95819 FM:: 0.95819 G-Mean:: 0.91878 Detection Rate: 0.95729 False Alarm Rate: 0.050223	Run=2 Accuracy is: 0.95245 Precision: 0.95245 Recall: 0.95245 FM:: 0.95245 G-Mean:: 0.90797 Detection Rate: 0.95068 False Alarm Rate: 0.057997	Run=2 Accuracy is: 0.96805 Precision: 0.96805 Recall: 0.96805 FM:: 0.96805 G-Mean:: 0.93747 Detection Rate: 0.96608 False Alarm Rate: 0.039647
Run =3 Accuracy is: 0.91347 Precision: 0.91347 Recall: 0.91347 FM:: 0.91347 G-Mean:: 0.83693 Detection Rate: 0.90824 False Alarm Rate: 0.11214	Run = 3 Accuracy is: 0.95775 Precision: 0.95775 Recall: 0.95775 FM:: 0.95775 G-Mean:: 0.91791 Detection Rate: 0.95496 False Alarm Rate: 0.052987	Run=3 Accuracy is: 0.94641 Precision: 0.94641 Recall: 0.94641 FM:: 0.94641 G-Mean:: 0.89669 Detection Rate: 0.94023 False Alarm Rate: 0.071029	Run=3 Accuracy is: 0.97161 Precision: 0.97161 Recall: 0.97161 FM:: 0.97161 G-Mean:: 0.94431 Detection Rate: 0.96586 False Alarm Rate: 0.040158
Run =4 Accuracy is: 0.90194 Precision: 0.90194 Recall: 0.90194 FM:: 0.90194 G-Mean:: 0.81675 Detection Rate: 0.93882 False Alarm Rate: 0.072564	Run = 4 Accuracy is: 0.96056 Precision: 0.96056 Recall: 0.96056 FM:: 0.96056 G-Mean:: 0.92321 Detection Rate: 0.95366 False Alarm Rate: 0.054795	Run=4 Accuracy is: 0.94965 Precision: 0.94965 Recall: 0.94965 FM:: 0.94965 G-Mean:: 0.9028 Detection Rate: 0.94659 False Alarm Rate: 0.063455	Run=4 Accuracy is: 0.96662 Precision: 0.96662 Recall: 0.96662 FM:: 0.96662 G-Mean:: 0.93474 Detection Rate: 0.95947 False Alarm Rate: 0.047835
Run=5 Accuracy is: 0.90447 Precision: 0.90447 Recall: 0.90447 FM:: 0.90447 G-Mean:: 0.82118 Detection Rate: 0.93965 False Alarm Rate: 0.070995	Run=5 Accuracy is: 0.95834 Precision: 0.95834 Recall: 0.95834 FM:: 0.95834 G-Mean:: 0.91908 Detection Rate: 0.96783 False Alarm Rate: 0.037156	Run=5 Accuracy is: 0.94732 Precision: 0.94732 Recall: 0.94732 FM:: 0.94732 G-Mean:: 0.89846 Detection Rate: 0.93887 False Alarm Rate: 0.073792	Run=5 Accuracy is: 0.96875 Precision: 0.96875 Recall: 0.96875 FM:: 0.96875 G-Mean:: 0.93881 Detection Rate: 0.96114 False Alarm Rate: 0.04589
Run=6	Run = 6	Run=6 Accuracy is: 0.93268	Run=6 Accuracy is: 0.97018

Accuracy is: 0.86759 Precision: 0.86759 Recall: 0.86759 FM:: 0.86759 G-Mean:: 0.75762 Detection Rate: 0.88139 False Alarm Rate: 0.13882	Accuracy is: 0.96096 Precision: 0.96096 Recall: 0.96096 FM:: 0.96096 G-Mean:: 0.92402 Detection Rate: 0.95678 False Alarm Rate: 0.051179	Precision: 0.93268 Recall: 0.93268 FM:: 0.93268 G-Mean:: 0.87155 Detection Rate: 0.93821 False Alarm Rate: 0.073246	Precision: 0.97018 Recall: 0.97018 FM:: 0.97018 G-Mean:: 0.94158 Detection Rate: 0.96926 False Alarm Rate: 0.035791
Run=7 Accuracy is: 0.91002 Precision: 0.91002 Recall: 0.91002 FM:: 0.91002 G-Mean:: 0.83073 Detection Rate: 0.89937 False Alarm Rate: 0.12442	Run=7 Accuracy is: 0.96402 Precision: 0.96402 Recall: 0.96402 FM:: 0.96402 G-Mean:: 0.9298 Detection Rate: 0.95682 False Alarm Rate: 0.051179	Run=7 Accuracy is: 0.94179 Precision: 0.94179 Recall: 0.94179 FM:: 0.94179 G-Mean:: 0.88817 Detection Rate: 0.93464 False Alarm Rate: 0.078432	Run=7 Accuracy is: 0.96621 Precision: 0.96621 Recall: 0.96621 FM:: 0.96621 G-Mean:: 0.93395 Detection Rate: 0.96149 False Alarm Rate: 0.04531
Run=8 Accuracy is: 0.90002 Precision: 0.90002 Recall: 0.90002 FM:: 0.90002 G-Mean:: 0.81332 Detection Rate: 0.91192 False Alarm Rate: 0.10433	Run=8 Accuracy is: 0.9593 Precision: 0.9593 Recall: 0.9593 FM:: 0.9593 G-Mean:: 0.92085 Detection Rate: 0.95413 False Alarm Rate: 0.054215	Run=8 Accuracy is: 0.94867 Precision: 0.94867 Recall: 0.94867 FM:: 0.94867 G-Mean:: 0.90091 Detection Rate: 0.93458 False Alarm Rate: 0.079251	Run=8 Accuracy is: 0.96478 Precision: 0.96478 Recall: 0.96478 FM:: 0.96478 G-Mean:: 0.93124 Detection Rate: 0.95894 False Alarm Rate: 0.048381
Run=9 Accuracy is: 0.88869 Precision: 0.88869 Recall: 0.88869 FM:: 0.88869 G-Mean:: 0.79388 Detection Rate: 0.8812 False Alarm Rate: 0.15151	Run=9 Accuracy is: 0.9574 Precision: 0.9574 Recall: 0.9574 FM:: 0.9574 G-Mean:: 0.91725 Detection Rate: 0.9617 False Alarm Rate: 0.044526	Run=9 Accuracy is: 0.95016 Precision: 0.95016 Recall: 0.95016 FM:: 0.95016 G-Mean:: 0.90379 Detection Rate: 0.95089 False Alarm Rate: 0.058338	Run=9 Accuracy is: 0.96513 Precision: 0.96513 Recall: 0.96513 FM:: 0.96513 G-Mean:: 0.93192 Detection Rate: 0.96197 False Alarm Rate: 0.044594
Run=10 Accuracy is: 0.8659 Precision: 0.8659 Recall: 0.8659 FM:: 0.8659 G-Mean:: 0.75521 Detection Rate: 0.90295 False Alarm Rate: 0.11221	Run=10 Accuracy is: 0.96242 Precision: 0.96242 Recall: 0.96242 FM:: 0.96242 G-Mean:: 0.92677 Detection Rate: 0.96343 False Alarm Rate: 0.04282	Run=10 Accuracy is: 0.94321 Precision: 0.94321 Recall: 0.94321 FM:: 0.94321 G-Mean:: 0.89081 Detection Rate: 0.94165 False Alarm Rate: 0.069289	Run=10 Accuracy is: 0.96242 Precision: 0.96242 Recall: 0.96242 FM:: 0.96242 G-Mean:: 0.92677 Detection Rate: 0.96343 False Alarm Rate: 0.04282
Run=11 Accuracy is: 0.87941 Precision: 0.87941 Recall: 0.87941 FM:: 0.87941 G-Mean:: 0.77771 Detection Rate: 0.88015 False Alarm Rate: 0.14663	Run=11 Accuracy is: 0.95821 Precision: 0.95821 Recall: 0.95821 FM:: 0.95821 G-Mean:: 0.91879 Detection Rate: 0.95723 False Alarm Rate: 0.050189	Run=11 Accuracy is: 0.95041 Testing Accuracy is: 0.95126 Precision: 0.95126 Recall: 0.95126 FM:: 0.95126 G-Mean:: 0.90574 Detection Rate: 0.94259 False Alarm Rate: 0.068538	Run=11 Accuracy is: 0.9697 Precision: 0.9697 Recall: 0.9697 FM:: 0.9697 G-Mean:: 0.94063 Detection Rate: 0.95729 False Alarm Rate: 0.050804

Run=12 Accuracy is: 0.92552 Precision: 0.92552 Recall: 0.92552 FM:: 0.92552 G-Mean:: 0.85828 Detection Rate: 0.90447 False Alarm Rate: 0.11797	Run=12 Accuracy is: 0.96373 Precision: 0.96373 Recall: 0.96373 FM:: 0.96373 G-Mean:: 0.92925 Detection Rate: 0.95602 False Alarm Rate: 0.052202	Run=12 Accuracy is: 0.94486 Precision: 0.94486 Recall: 0.94486 FM:: 0.94486 G-Mean:: 0.89374 Detection Rate: 0.92808 False Alarm Rate: 0.087439	Run=12 Accuracy is: 0.96453 Precision: 0.96453 Recall: 0.96453 FM:: 0.96453 G-Mean:: 0.93079 Detection Rate: 0.95997 False Alarm Rate: 0.047323
Run=13 Accuracy is: 0.87492 Precision: 0.87492 Recall: 0.87492 FM:: 0.87492 G-Mean:: 0.76982 Detection Rate: 0.84333 False Alarm Rate: 0.20592	Run=13 Accuracy is: 0.95738 Precision: 0.95738 Recall: 0.95738 FM:: 0.95738 G-Mean:: 0.91724 Detection Rate: 0.95785 False Alarm Rate: 0.049336	Run=13 Accuracy is: 0.94896 Precision: 0.94896 Recall: 0.94896 FM:: 0.94896 G-Mean:: 0.90146 Detection Rate: 0.94117 False Alarm Rate: 0.070585	Run=13 Accuracy is: 0.97382 Precision: 0.97382 Recall: 0.97382 FM:: 0.97382 G-Mean:: 0.94857 Detection Rate: 0.97024 False Alarm Rate: 0.03487
Run=14 Accuracy is: 0.88991 Precision: 0.88991 Recall: 0.88991 FM:: 0.88991 G-Mean:: 0.79585 Detection Rate: 0.87028 False Alarm Rate: 0.16857	Run=14 Accuracy is: 0.95789 Precision: 0.95789 Recall: 0.95789 FM:: 0.95789 G-Mean:: 0.91822 Detection Rate: 0.95773 False Alarm Rate: 0.049678	Run=14 Accuracy is: 0.88991 Precision: 0.88991 Recall: 0.88991 FM:: 0.88991 G-Mean:: 0.79585 Detection Rate: 0.87028 False Alarm Rate: 0.16857	Run=14 Run=15Accuracy is: 0.95789 Precision: 0.95789 Recall: 0.95789 FM:: 0.95789 G-Mean:: 0.91822 Detection Rate: 0.95773 False Alarm Rate: 0.049678
Run=15 Accuracy is: 0.89842 Precision: 0.89842 Recall: 0.89842 FM:: 0.89842 G-Mean:: 0.81042 Detection Rate: 0.92371 False Alarm Rate: 0.08853	Run=15 Accuracy is: 0.96356 Precision: 0.96356 Recall: 0.96356 FM:: 0.96356 G-Mean:: 0.9289 Detection Rate: 0.95865 False Alarm Rate: 0.048654	Run=15 Accuracy is: 0.95789 Precision: 0.95789 Recall: 0.95789 FM:: 0.95789 G-Mean:: 0.91822 Detection Rate: 0.95773 False Alarm Rate: 0.049678	Run=15 Testing Accuracy is: 0.96356 Precision: 0.96356 Recall: 0.96356 FM:: 0.96356 G-Mean:: 0.9289 Detection Rate: 0.95865 False Alarm Rate: 0.048654

35 number of neurons		**50 number of neurons**	
ELM	**FLN**	**ELM**	**FLN**
Run 1 Accuracy is: 0.90242 Precision: 0.90242 Recall: 0.90242 FM:: 0.90242 G-Mean:: 0.81746 Detection Rate: 0.90128 False Alarm Rate: 0.11995	Run 1 Accuracy is: 0.95781 Precision: 0.95781 Recall: 0.95781 FM:: 0.95781 G-Mean:: 0.91807 Detection Rate: 0.96 False Alarm Rate: 0.046812	Run=1 Testing Accuracy is: 0.94873 Precision: 0.94873 Recall: 0.94873 FM:: 0.94873 G-Mean:: 0.90104 Detection Rate: 0.93925 False Alarm Rate: 0.073042	Run = 1 Testing Accuracy is: 0.96731 Precision: 0.96731 Recall: 0.96731 FM:: 0.96731 G-Mean:: 0.93605 Detection Rate: 0.96223 False Alarm Rate: 0.044355
Run =2 Accuracy is: 0.91574 Precision: 0.91574 Recall: 0.91574	Run =2 Accuracy is: 0.95819 Precision: 0.95819 Recall: 0.95819	Run=2 Accuracy is: 0.95245 Precision: 0.95245 Recall: 0.95245	Run=2 Accuracy is: 0.96805 Precision: 0.96805 Recall: 0.96805

FM:: 0.91574 G-Mean:: 0.84102 Detection Rate: 0.91703 False Alarm Rate: 0.1001	FM:: 0.95819 G-Mean:: 0.91878 Detection Rate: 0.95729 False Alarm Rate: 0.050223	FM:: 0.95245 G-Mean:: 0.90797 Detection Rate: 0.95068 False Alarm Rate: 0.057997	FM:: 0.96805 G-Mean:: 0.93747 Detection Rate: 0.96608 False Alarm Rate: 0.039647
Run =3 Accuracy is: 0.91347 Precision: 0.91347 Recall: 0.91347 FM:: 0.91347 G-Mean:: 0.83693 Detection Rate: 0.90824 False Alarm Rate: 0.11214	Run = 3 Accuracy is: 0.95775 Precision: 0.95775 Recall: 0.95775 FM:: 0.95775 G-Mean:: 0.91791 Detection Rate: 0.95496 False Alarm Rate: 0.052987	Run=3 Accuracy is: 0.94641 Precision: 0.94641 Recall: 0.94641 FM:: 0.94641 G-Mean:: 0.89669 Detection Rate: 0.94023 False Alarm Rate: 0.071029	Run=3 Accuracy is: 0.97161 Precision: 0.97161 Recall: 0.97161 FM:: 0.97161 G-Mean:: 0.94431 Detection Rate: 0.96586 False Alarm Rate: 0.040158
Run =4 Accuracy is: 0.90194 Precision: 0.90194 Recall: 0.90194 FM:: 0.90194 G-Mean:: 0.81675 Detection Rate: 0.93882 False Alarm Rate: 0.072564	Run = 4 Accuracy is: 0.96056 Precision: 0.96056 Recall: 0.96056 FM:: 0.96056 G-Mean:: 0.92321 Detection Rate: 0.95366 False Alarm Rate: 0.054795	Run=4 Accuracy is: 0.94965 Precision: 0.94965 Recall: 0.94965 FM:: 0.94965 G-Mean:: 0.9028 Detection Rate: 0.94659 False Alarm Rate: 0.063455	Run=4 Accuracy is: 0.96662 Precision: 0.96662 Recall: 0.96662 FM:: 0.96662 G-Mean:: 0.93474 Detection Rate: 0.95947 False Alarm Rate: 0.047835
Run=5 Accuracy is: 0.90447 Precision: 0.90447 Recall: 0.90447 FM:: 0.90447 G-Mean:: 0.82118 Detection Rate: 0.93965 False Alarm Rate: 0.070995	Run=5 Accuracy is: 0.95834 Precision: 0.95834 Recall: 0.95834 FM:: 0.95834 G-Mean:: 0.91908 Detection Rate: 0.96783 False Alarm Rate: 0.037156	Run=5 Accuracy is: 0.94732 Precision: 0.94732 Recall: 0.94732 FM:: 0.94732 G-Mean:: 0.89846 Detection Rate: 0.93887 False Alarm Rate: 0.073792	Run=5 Accuracy is: 0.96875 Precision: 0.96875 Recall: 0.96875 FM:: 0.96875 G-Mean:: 0.93881 Detection Rate: 0.96114 False Alarm Rate: 0.04589
Run=6 Accuracy is: 0.86759 Precision: 0.86759 Recall: 0.86759 FM:: 0.86759 G-Mean:: 0.75762 Detection Rate: 0.88139 False Alarm Rate: 0.13882	Run = 6 Accuracy is: 0.96096 Precision: 0.96096 Recall: 0.96096 FM:: 0.96096 G-Mean:: 0.92402 Detection Rate: 0.95678 False Alarm Rate: 0.051179	Run=6 Accuracy is: 0.93268 Precision: 0.93268 Recall: 0.93268 FM:: 0.93268 G-Mean:: 0.87155 Detection Rate: 0.93821 False Alarm Rate: 0.073246	Run=6 Accuracy is: 0.97018 Precision: 0.97018 Recall: 0.97018 FM:: 0.97018 G-Mean:: 0.94158 Detection Rate: 0.96926 False Alarm Rate: 0.035791
Run=7 Accuracy is: 0.91002 Precision: 0.91002 Recall: 0.91002 FM:: 0.91002 G-Mean:: 0.83073 Detection Rate: 0.89937 False Alarm Rate: 0.12442	Run=7 Accuracy is: 0.96402 Precision: 0.96402 Recall: 0.96402 FM:: 0.96402 G-Mean:: 0.9298 Detection Rate: 0.95682 False Alarm Rate: 0.051179	Run=7 Accuracy is: 0.94179 Precision: 0.94179 Recall: 0.94179 FM:: 0.94179 G-Mean:: 0.88817 Detection Rate: 0.93464 False Alarm Rate: 0.078432	Run=7 Accuracy is: 0.96621 Precision: 0.96621 Recall: 0.96621 FM:: 0.96621 G-Mean:: 0.93395 Detection Rate: 0.96149 False Alarm Rate: 0.04531
Run=8 Accuracy is: 0.90002 Precision: 0.90002	Run=8 Accuracy is: 0.9593 Precision: 0.9593	Run=8 Accuracy is: 0.94867 Precision: 0.94867	Run=8 Accuracy is: 0.96478 Precision: 0.96478

Recall: 0.90002 FM:: 0.90002 G-Mean:: 0.81332 Detection Rate: 0.91192 False Alarm Rate: 0.10433	Recall: 0.9593 FM:: 0.9593 G-Mean:: 0.92085 Detection Rate: 0.95413 False Alarm Rate: 0.054215	Recall: 0.94867 FM:: 0.94867 G-Mean:: 0.90091 Detection Rate: 0.93458 False Alarm Rate: 0.079251	Recall: 0.96478 FM:: 0.96478 G-Mean:: 0.93124 Detection Rate: 0.95894 False Alarm Rate: 0.048381
Run=9 Accuracy is: 0.88869 Precision: 0.88869 Recall: 0.88869 FM:: 0.88869 G-Mean:: 0.79388 Detection Rate: 0.8812 False Alarm Rate: 0.15151	Run=9 Accuracy is: 0.9574 Precision: 0.9574 Recall: 0.9574 FM:: 0.9574 G-Mean:: 0.91725 Detection Rate: 0.9617 False Alarm Rate: 0.044526	Run=9 Accuracy is: 0.95016 Precision: 0.95016 Recall: 0.95016 FM:: 0.95016 G-Mean:: 0.90379 Detection Rate: 0.95089 False Alarm Rate: 0.058338	Run=9 Accuracy is: 0.96513 Precision: 0.96513 Recall: 0.96513 FM:: 0.96513 G-Mean:: 0.93192 Detection Rate: 0.96197 False Alarm Rate: 0.044594
Run=10 Accuracy is: 0.8659 Precision: 0.8659 Recall: 0.8659 FM:: 0.8659 G-Mean:: 0.75521 Detection Rate: 0.90295 False Alarm Rate: 0.11221	Run=10 Accuracy is: 0.96242 Precision: 0.96242 Recall: 0.96242 FM:: 0.96242 G-Mean:: 0.92677 Detection Rate: 0.96343 False Alarm Rate: 0.04282	Run=10 Accuracy is: 0.94321 Precision: 0.94321 Recall: 0.94321 FM:: 0.94321 G-Mean:: 0.89081 Detection Rate: 0.94165 False Alarm Rate: 0.069289	Run=10 Accuracy is: 0.96242 Precision: 0.96242 Recall: 0.96242 FM:: 0.96242 G-Mean:: 0.92677 Detection Rate: 0.96343 False Alarm Rate: 0.04282
Run=11 Accuracy is: 0.87941 Precision: 0.87941 Recall: 0.87941 FM:: 0.87941 G-Mean:: 0.77771 Detection Rate: 0.88015 False Alarm Rate: 0.14663	Run=11 Accuracy is: 0.95821 Precision: 0.95821 Recall: 0.95821 FM:: 0.95821 G-Mean:: 0.91879 Detection Rate: 0.95723 False Alarm Rate: 0.050189	Run=11 Accuracy is: 0.95041 Testing Accuracy is: 0.95126 Precision: 0.95126 Recall: 0.95126 FM:: 0.95126 G-Mean:: 0.90574 Detection Rate: 0.94259 False Alarm Rate: 0.068538	Run=11 Accuracy is: 0.9697 Precision: 0.9697 Recall: 0.9697 FM:: 0.9697 G-Mean:: 0.94063 Detection Rate: 0.95729 False Alarm Rate: 0.050804
Run=12 Accuracy is: 0.92552 Precision: 0.92552 Recall: 0.92552 FM:: 0.92552 G-Mean:: 0.85828 Detection Rate: 0.90447 False Alarm Rate: 0.11797	Run=12 Accuracy is: 0.96373 Precision: 0.96373 Recall: 0.96373 FM:: 0.96373 G-Mean:: 0.92925 Detection Rate: 0.95602 False Alarm Rate: 0.052202	Run=12 Accuracy is: 0.94486 Precision: 0.94486 Recall: 0.94486 FM:: 0.94486 G-Mean:: 0.89374 Detection Rate: 0.92808 False Alarm Rate: 0.087439	Run=12 Accuracy is: 0.96453 Precision: 0.96453 Recall: 0.96453 FM:: 0.96453 G-Mean:: 0.93079 Detection Rate: 0.95997 False Alarm Rate: 0.047323
Run=13 Accuracy is: 0.87492 Precision: 0.87492 Recall: 0.87492 FM:: 0.87492 G-Mean:: 0.76982 Detection Rate: 0.84333 False Alarm Rate: 0.20592	Run=13 Accuracy is: 0.95738 Precision: 0.95738 Recall: 0.95738 FM:: 0.95738 G-Mean:: 0.91724 Detection Rate: 0.95785 False Alarm Rate: 0.049336	Run=13 Accuracy is: 0.94896 Precision: 0.94896 Recall: 0.94896 FM:: 0.94896 G-Mean:: 0.90146 Detection Rate: 0.94117 False Alarm Rate: 0.070585	Run=13 Accuracy is: 0.97382 Precision: 0.97382 Recall: 0.97382 FM:: 0.97382 G-Mean:: 0.94857 Detection Rate: 0.97024 False Alarm Rate: 0.03487
Run=14 Accuracy is: 0.88991	Run=14 Accuracy is: 0.95789	Run=14 Accuracy is: 0.88991	Run=14 Run=15Accuracy is: 0.95789

Precision: 0.88991 Recall: 0.88991 FM:: 0.88991 G-Mean:: 0.79585 Detection Rate: 0.87028 False Alarm Rate: 0.16857	Precision: 0.95789 Recall: 0.95789 FM:: 0.95789 G-Mean:: 0.91822 Detection Rate: 0.95773 False Alarm Rate: 0.049678	Precision: 0.88991 Recall: 0.88991 FM:: 0.88991 G-Mean:: 0.79585 Detection Rate: 0.87028 False Alarm Rate: 0.16857	Precision: 0.95789 Recall: 0.95789 FM:: 0.95789 G-Mean:: 0.91822 Detection Rate: 0.95773 False Alarm Rate: 0.049678
Run=15 Accuracy is: 0.89842 Precision: 0.89842 Recall: 0.89842 FM:: 0.89842 G-Mean:: 0.81042 Detection Rate: 0.92371 False Alarm Rate: 0.08853	Run=15 Accuracy is: 0.96356 Precision: 0.96356 Recall: 0.96356 FM:: 0.96356 G-Mean:: 0.9289 Detection Rate: 0.95865 False Alarm Rate: 0.048654	Run=15 Accuracy is: 0.95789 Precision: 0.95789 Recall: 0.95789 FM:: 0.95789 G-Mean:: 0.91822 Detection Rate: 0.95773 False Alarm Rate: 0.049678	Run=15 Testing Accuracy is: 0.96356 Precision: 0.96356 Recall: 0.96356 FM:: 0.96356 G-Mean:: 0.9289 Detection Rate: 0.95865 False Alarm Rate: 0.048654

APPENDIX B
EXPERIMENTS RESULTS OF PSO-FLN WITH SEVERAL RUNS

P=10-Itr=100-Neurons=10

Precision	Recall	F_Measure	G_Mean	PSO.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9679	0.9786	0.9732	0.9712	0.9869	0.9857	0.9892	0.98712	0.9679	0.03725
0.9701	0.9739	0.9721	0.9701	0.9857				0.9701	0.03442
0.9623	0.9798	0.9711	0.9687	0.9859				0.9623	0.02204
0.9663	0.9799	0.9731	0.9711	0.9867				0.9663	0.03916
0.9751	0.9743	0.9729	0.9729	0.9866				0.9751	0.02865
0.9678	0.9884	0.9781	0.9762	0.9889				0.9678	0.03773
0.9854	0.9714	0.9734	0.9716	0.9859				0.9754	0.0281
0.9767	0.9762	0.9765	0.9748	0.9881				0.9767	0.0267
0.9726	0.9744	0.9735	0.9717	0.9865				0.9726	0.0314
0.9711	0.9809	0.9761	0.9742	0.9881				0.9711	0.0335
0.9703	0.9748	0.9794	0.9731	0.9869				0.9703	0.03442
0.9786	0.9777	0.9782	0.9776	0.9892				0.9786	0.02452
0.9702	0.9783	0.9783	0.9722	0.9867				0.9702	0.0347
0.9679	0.9837	0.9757	0.9739	0.9868				0.9679	0.0373
0.9721	0.9803	0.9762	0.9744	0.9879				0.9721	0.0323
0.971627	0.978173	0.975187	0.972913					0.97096	0.032146

P=10-Itr=250-Neurons=10

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9658	0.9874	0.9765	0.9746	0.9888	0.9879	0.9929	0.989407	0.9658	0.0401
0.9728	0.9896	0.9811	0.9797	0.9901				0.9728	0.0317
0.9771	0.9882	0.9826	0.9813	0.9912				0.9912	0.0265
0.9816	0.9902	0.9857	0.9847	0.9929				0.9816	0.0213
0.9728	0.9867	0.9797	0.9781	0.9903				0.9728	0.0315
0.9687	0.9842	0.9764	0.9745	0.9887				0.9687	0.0365
0.9808	0.9885	0.9846	0.9835	0.9916				0.9808	0.0221
0.9689	0.9903	0.9795	0.9778	0.9901				0.9689	0.0364
0.9628	0.9869	0.9747	0.9726	0.9879				0.9628	0.0438
0.9754	0.99	0.9827	0.9813	0.9912				0.9754	0.0286
0.9768	0.9837	0.9802	0.9788	0.9907				0.9768	0.0268
0.9835	0.9898	0.9866	0.9856	0.9929				0.9835	0.019
0.9772	0.9824	0.9798	0.9783	0.9902				0.9772	0.0263
0.9749	0.9823	0.9786	0.977	0.9897				0.9749	0.0289
0.9663	0.9877	0.9769	0.975	0.9885				0.9663	0.0395

P=10-Itr500-Neurons=10

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9713	0.9788	0.9751	0.9732	0.9881	0.9871	0.9931	0.99032	0.9713	0.03312
0.9769	0.9755	0.9762	0.9746	0.9879				0.9769	0.02644
0.9785	0.9827	0.9806	0.9792	0.9897				0.9785	0.02473
0.9722	0.9855	0.9788	0.9772	0.9896				0.9722	0.03227
0.9798	0.9792	0.9795	0.9781	0.9902				0.9798	0.02313
0.9766	0.9797	0.9782	0.9766	0.9893				0.9766	0.0269
0.9711	0.9834	0.9766	0.9748	0.9873				0.9711	0.03493
0.9754	0.99	0.9827	0.9813	0.9912				0.9754	0.0286
0.9731	0.9777	0.9754	0.9736	0.9931				0.9731	0.03101
0.9798	0.9792	0.9795	0.9781	0.9902				0.9798	0.02313
0.9766	0.9797	0.9782	0.9766	0.9893				0.9766	0.0269
0.9711	0.9834	0.9766	0.9748	0.9873				0.9711	0.03493
0.9643	0.9838	0.9739	0.9718	0.9871				0.9643	0.0418
0.9731	0.9777	0.9754	0.9736	0.9879				0.9731	0.03101
0.9835	0.9898	0.9866	0.9856	0.9929				0.9835	0.019
0.97488	0.98174	0.97822	0.976607					0.974887	0.029193

P=10-
Itr100-
Neurons=25

Precision	Recall	Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9658	0.9874	0.9765	0.9746	0.9888	0.9879	0.9929	0.990194	0.9658	0.0401
0.9728	0.9896	0.9811	0.9797	0.9901				0.9728	0.0317
0.9771	0.9882	0.9826	0.9813	0.9912				0.9771	0.0265
0.9816	0.9901	0.9857	0.9847	0.9929				0.9816	0.0213
0.9728	0.9867	0.9797	0.9781	0.9903				0.9728	0.0315
0.9687	0.9842	0.9764	0.9745	0.9887				0.9687	0.0365
0.9808	0.9885	0.9846	0.9835	0.9916				0.9808	0.0221
0.9689	0.9903	0.9795	0.9778	0.9905				0.9689	0.0364
0.9628	0.9869	0.9747	0.9726	0.9879				0.9628	0.0438
0.9628	0.9869	0.9747	0.9726	0.9879				0.9628	0.0438
0.9754	0.9901	0.9827	0.9813	0.9912				0.9754	0.0286
0.9768	0.9837	0.9802	0.9788	0.9907				0.9768	0.0268
0.9835	0.9898	0.9866	0.9856	0.9929				0.9835	0.019
0.9772	0.9824	0.9798	0.9783	0.9902				0.9772	0.0263
0.9749	0.9823	0.9786	0.9771	0.9897				0.9749	0.0289
0.9663	0.9877	0.9769	0.975	0.9885				0.9663	0.0395

P=10-
Itr250
Neurons=25

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9774	0.9845	0.9809	0.9796	0.9907	0.9894	0.9938	0.991	0.9774	0.02613
0.9925	0.9844	0.9884	0.9844	0.9938				0.9925	0.0084
0.9742	0.9841	0.9777	0.9761	0.9894				0.9742	0.0298
0.9741	0.9861	0.9801	0.9785	0.9904				0.9741	0.0301
0.9774	0.9893	0.9833	0.9821	0.9916				0.9774	0.0262
0.9746	0.9832	0.9789	0.9773	0.9896				0.9746	0.0294
0.9984	0.9891	0.9867	0.9857	0.9926				0.9844	0.0179

0.9791	0.9901	0.9845	0.9833	0.9925				0.9791	0.0242
0.9752	0.9839	0.9795	0.9781	0.9903				0.9752	0.0286
0.9758	0.9851	0.9808	0.9791	0.9906				0.9758	0.0281
0.9751	0.9893	0.9822	0.9808	0.9911				0.9755	0.0289
0.9749	0.9894	0.9821	0.9807	0.9909				0.9749	0.0292
0.9754	0.9866	0.9811	0.9795	0.9907				0.9754	0.0285
0.9772	0.9811	0.9791	0.9772	0.9895				0.9772	0.0262
0.9711	0.9912	0.9812	0.9795	0.9907				0.9711	0.0338
0.9835	0.9898	0.9866	0.9856	0.9929				0.9835	0.019

P=10-
Itr500
Neurons=25

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9714	0.9891	0.9801	0.9786	0.9906	0.9899	0.9931	0.9921	0.9716	0.0333
0.99838	0.9826	0.9832	0.0.9821	0.9917				0.9839	0.0185
0.9748	0.9891	0.9819	0.9805	0.9909				0.9749	0.093
0.9778	0.9821	0.9799	0.9785	0.9899				0.9778	0.0255
0.9821	0.9832	0.9855	0.9846	0.9931				0.9879	0.0138
0.9829	0.9779	0.9805	0.9791	0.9908				0.9828	0.0194
0.9783	0.9845	0.9814	0.9807	0.9911				0.9785	0.025
0.9765	0.9901	0.9832	0.982	0.9917				0.9766	0.0273
0.9781	0.9839	0.9811	0.9769	0.9906				0.9783	0.0253
0.9822	0.9851	0.9836	0.9824	0.9919				0.9823	0.0204
0.9821	0.9878	0.9849	0.9838	0.9921				0.9822	0.0207
0.9778	0.9849	0.9813	0.9799	0.9909				0.9777	0.0257
0.9817	0.9832	0.9855	0.9846	0.9931				0.9879	0.0138
0.9829	0.9779	0.9805	0.9791	0.9908				0.9828	0.0194

**P=10-
Itr100
Neurons=35**

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9777	0.9896	0.9836	0.9824	0.9918	0.9897	0.9929	0.99136	0.9777	0.0258
0.9754	0.9891	0.9822	0.9809	0.9913				0.9754	0.2862
0.9804	0.9895	0.9849	0.9838	0.9921				0.9804	0.0226
0.9749	0.9903	0.9826	0.9812	0.9916				0.9749	0.0292
0.9834	0.9844	0.9838	0.9827	0.9921				0.9834	0.019
0.9777	0.9927	0.9851	0.9841	0.9923				0.9777	0.0259
0.9824	0.9881	0.9853	0.9842	0.9929				0.9824	0.0202
0.9774	0.9881	0.9827	0.9814	0.9912				0.9774	0.0261
0.9738	0.9858	0.9798	0.9782	0.9901				0.9738	0.0301
0.9732	0.9879	0.9805	0.9791	0.9905				0.9732	0.0312
0.9814	0.9825	0.9819	0.9807	0.9915				0.9814	0.0213
0.9704	0.9915	0.9809	0.9793	0.9907				0.9704	0.0346
0.9787	0.9899	0.9843	0.9831	0.9923				0.9787	0.0247
0.9687	0.9895	0.9791	0.9773	0.9897				0.9687	0.0366
0.9687	0.9908	0.9796	0.9781	0.9903				0.9687	0.0366

**P=10-
Itr250
Neurons=35**

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9819	0.9878	0.9849	0.9838	0.9922	0.9908	0.995	0.9925	0.9819	0.0208
0.9821	0.9923	0.9872	0.9863	0.9932				0.9819	0.0209
0.9774	0.9918	0.9845	0.9833	0.9927				0.9774	0.0263
0.9842	0.9864	0.9843	0.9843	0.9925				0.9842	0.0181
0.9752	0.9869	0.9811	0.9796	0.9908				0.9752	0.0287
0.9786	0.9904	0.9845	0.9833	0.9925				0.9786	0.0248
0.9831	0.9841	0.9836	0.9824	0.9921				0.9831	0.0193
0.9798	0.9874	0.9836	0.9824	0.9921				0.9798	0.0233
0.9802	0.9899	0.9851	0.9839	0.9925				0.9801	0.0231
0.9839	0.9918	0.9877	0.9868	0.9939				0.9836	0.0189
0.9746	0.9919	0.9832	0.9818	0.9918				0.9746	0.0296
0.9906	0.9901	0.9904	0.9897	0.9952				0.9906	0.0107
0.9854	0.9898	0.9876	0.9867	0.9939				0.9854	0.0167
0.9777	0.9856	0.9816	0.9803	0.9911				0.9777	0.0257
0.9775	0.9864	0.9821	0.9807	0.9914				0.9776	0.0258

P=10-Itr500 Neurons=35

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9819	0.9878	0.9849	0.9838	0.9924	0.9903	0.9944	0.9928	0.9821	0.0208
0.9864	0.9866	0.9865	0.9934	0.9934				0.9864	0.0156
0.9787	0.9866	0.9826	0.9813	0.9916				0.9877	0.0246
0.9874	0.9895	0.9884	0.9876	0.9944				0.9875	0.0144
0.9743	0.9851	0.9797	0.9781	0.9903				0.9745	0.0971
0.9812	0.9897	0.9855	0.9844	0.9929				0.9813	0.0216
0.9836	0.9878	0.9857	0.9847	0.9932				0.9836	0.0189
0.9851	0.9908	0.9879	0.9871	0.9939				0.9851	0.0171
0.9801	0.9873	0.9837	0.9825	0.9923				0.9801	0.0229
0.9831	0.9841	0.9836	0.9824	0.9921				0.9831	0.0193
0.9798	0.9874	0.9836	0.9824	0.9921				0.9798	0.0233
0.9802	0.9899	0.9851	0.9839	0.9925				0.9801	0.0231
0.9839	0.9918	0.9877	0.9868	0.9939				0.9836	0.0189
0.9836	0.9878	0.9857	0.9847	0.9932				0.9836	0.0189
0.9851	0.9908	0.9879	0.9871	0.9939				0.9851	0.0171

P=10-Itr100 Neurons=50

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9819	0.9912	0.9865	0.9855	0.9931	0.9919	0.9945	0.993267	0.9819	0.0209
0.9837	0.9905	0.9871	0.9861	0.9935				0.9837	0.0187
0.9774	0.9895	0.9843	0.9822	0.9919				0.9774	0.0262
0.9869	0.9879	0.9874	0.9865	0.9937				0.9869	0.015
0.9849	0.9917	0.9883	0.9875	0.9941				0.9849	0.0173
0.9826	0.9911	0.9868	0.9858	0.9932				0.9826	0.021
0.9892	0.9907	0.9901	0.9893	0.9945				0.9892	0.0123
0.9824	0.9915	0.9869	0.9861	0.9937				0.9824	0.0203

0.9792	0.9909	0.9851	0.9839	0.9925				0.9792	0.024
0.9835	0.9918	0.9877	0.9868	0.9941				0.9835	0.019
0.9864	0.9865	0.9865	0.9856	0.9932				0.9864	0.0155
0.9773	0.9916	0.9844	0.9832	0.9922				0.9773	0.0263
0.9853	0.9908	0.9881	0.9872	0.9943				0.9853	0.0168
0.9825	0.9861	0.9844	0.9833	0.9925				0.9825	0.02
0.9832	0.9867	0.9849	0.9839	0.9925				0.9832	0.0193
0.98309	0.9899	0.98656	0.9855					0.983093	0.019507

P=10-
Itr250
Neurons=50

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9848	0.9902	0.9875	0.9866	0.9936	0.993981	0.9951	0.993981	0.9848	0.0175
0.9851	0.9898	0.9875	0.9866	0.9935				0.9851	0.0171
0.9867	0.9891	0.9878	0.9871	0.9941				0.9867	0.0153
0.9914	0.9892	0.9903	0.9897	0.9951				0.9914	0.0098
0.9899	0.9889	0.9894	0.9887	0.9948				0.9899	0.0115
0.9883	0.9901	0.9892	0.9884	0.9946				0.9883	0.0134
0.9848	0.9914	0.9881	0.9872	0.9941				0.9848	0.0175
0.9906	0.9891	0.9898	0.9891	0.9949				0.9906	0.0107
0.9867	0.9891	0.9879	0.9871	0.9941				0.9867	0.0152
0.9801	0.9902	0.9851	0.9841	0.9927				0.9801	0.0231
0.9868	0.9874	0.9871	0.9862	0.9936				0.9868	0.0151
0.9889	0.9901	0.9895	0.9887	0.9948				0.9889	0.0126
0.9861	0.9886	0.9873	0.9865	0.9939				0.9861	0.016
0.9801	0.9902	0.9851	0.9841	0.9927				0.9801	0.0231
0.9868	0.9874	0.9871	0.9862	0.9936				0.9868	0.0151
0.9848	0.9902	0.9875	0.9866	0.9936				0.9848	0.0175
0.98639	0.989438	0.987888	0.987056					0.986369	0.015656

P=10-
Itr500
Neurons=50

Precision	Recall	F_Measure	G_Mean	pso.acc	Min.acc	Max.acc	average.acc	DR	FAR
0.9932	0.9886	0.9911	0.9904	0.9952	0.9933	0.9955	0.994631	0.9933	0.0077
0.9852	0.9891	0.9875	0.9875	0.9939				0.9855	0.017
0.9912	0.9908	0.9911	0.9903	0.9954				0.9914	0.01
0.9845	0.9899	0.9873	0.9863	0.9933				0.9846	0.0178
0.9883	0.9915	0.9898	0.9891	0.9951				0.9885	0.0135
0.9865	0.9902	0.9884	0.9875	0.9941				0.9866	0.0155
0.9843	0.9906	0.9875	0.9866	0.9942				0.9845	0.0181
0.9882	0.9927	0.9904	0.9897	0.9952				0.9886	0.0136
0.9881	0.9913	0.9897	0.9889	0.9949				0.9882	0.0137
0.9878	0.9916	0.9897	0.9891	0.9953				0.9879	0.0141
0.9906	0.9912	0.9909	0.9903	0.9953				0.9909	0.0104
0.9903	0.9913	0.9908	0.9901	0.9955				0.9905	0.0111

0.9845	0.9899	0.9873	0.9863	0.9933	0.9846	0.0178
0.9883	0.9915	0.9898	0.9891	0.9951	0.9885	0.0135
0.9865	0.9902	0.9884	0.9875	0.9941	0.9866	0.0155
0.9843	0.9906	0.9875	0.9866	0.9942	0.9845	0.0181
0.98761	0.990688	0.9892	0.988456		0.987794	0.014213

List of Publication

1. Ali, Mohammed Hasan, et al. "**A new intrusion detection system based on Fast Learning Network and Particle swarm optimization**." ***IEEE Access*** 6 (2018): 20255-20261.ISI

2. Mohammed Hasan, Ali, and Zolkipli Mohamad Fadli. "**Propose Intrusion-Detection System Based on Fast Learning Network in Cloud Computing**." (2017): 1-1. The 5th International Conference on Software Engineering & Computer System (ICSECS' 17),

3. Mohammed Hasan Ali, Mohamad Fadli Zolkipli, et al, 2017. **Intrusion Detection System Based on Machine Learning in Cloud Computing**. *Journal of Engineering and Applied Sciences, 12: 4241-4245*

4. Mohammed Hasan Ali, and Mohamad Fadli Zolkipli. "**Review On Hybrid Extreme Learning Machine and Genetic Algorithm To Work As Intrusion Detection System**.". ARPN Journal, VOL. 11, NO. 1, JANUARY 2016

5. Hussam Alddin S. Ahmed, Mohammed Hasan Ali, et al "**A Review of Challenges and Security Risks of Cloud Computing**". *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 9.1-2 (2017): 87-91.

6. Mohammed Hasan Ali, Mohamad Fadli Zolkipli, et al, 2017. "**Enhance of Extreme Learning Machine-Genetic Algorithm Hybrid Based on Intrusion Detection System**". Journal of Engineering and Applied Sciences, 12: 4180-4185.

7. Mohammed Hasan Ali, Mohamad Fadli Zolkipli," **Model of Improved a Kernel Fast Learning Network based on Intrusion Detection System**" has been accepted for presentation at the ICO'2018 conference and for publication in the conference proceedings published by SPRINGER,2018