Automatic Building of a Powerful IDS for The Cloud Based on Deep Neural Network by Using a Novel Combination of Simulated Annealing Algorithm and Improved Self-Adaptive Genetic Algorithm

Zouhair Chiba[1], Moulay Seddiq El Kasmi Alaoui[1], Noreddine Abghour[1] and Khalid Moussaid[1]

[1]LIS Labs, Faculty of Sciences Aïn Chock, Hassan II University of Casablanca, Morocco

***Abstract*:** Cloud computing (CC) is the fastest-growing data hosting and computational technology that stands today as a satisfactory answer to the problem of data storage and computing. Thereby, most organizations are now migrating their services into the cloud due to its appealing features and its tangible advantages. Nevertheless, providing privacy and security to protect cloud assets and resources still a very challenging issue. To address the above issues, we propose a smart approach to construct automatically an efficient and effective anomaly network IDS based on Deep Neural Network, by using a novel hybrid optimization framework "ISAGASAA". ISAGASAA framework combines our new self-adaptive heuristic search algorithm called "Improved Self-Adaptive Genetic Algorithm" (ISAGA) and Simulated Annealing Algorithm (SAA). Our approach consists of using ISAGASAA with the aim of seeking the optimal or near optimal combination of most pertinent values of the parameters included in building of DNN based IDS or impacting its performance, which guarantee high detection rate, high accuracy and low false alarm rate. The experimental results turn out the capability of our IDS to uncover intrusions with high detection accuracy and low false alarm rate, and demonstrate its superiority in comparison with state-of-the-art methods.

***Keywords*:** Cloud computing, Network intrusion detection system, Deep neural network, Genetic algorithm, Self-adaptive heuristic search algorithm, Simulated annealing algorithm.

1. Introduction

At present, cloud computing has become an irreversible service trend. In fact, most organizations are now migrating their services into the cloud to offer a more flexible, open, mobile and ubiquitous service [1]. There are several definitions of Cloud computing (CC), but the most popular is that of the NIST organization (National Institute of Standards and Technology). According to NIST, CC is a model of IT that delivers convenient, on-demand network access to a shared pool of configurable computing resources (i.e., networks, servers, storage, applications, etc.) "as service" over the internet, for satisfying computing demand of users, on pay as use basis. Provision and releasing of resources are done by service providers with minimal effort [2]. NIST introduces CC by considering its 5 main features (i.e., bandwidth, rapid flexibility, measurable, on-demand service, and resource pooling), its 3 service delivering models (i.e., software as a service (SaaS) such as Microsoft's Azure, platform as a service (PaaS) such as Google App Engine, and infrastructure as a service (IaaS) such as Google apps) [3] and its 3 deployment models (i.e., public, private and hybrid) [4]. CC is a distributed model which supplies computing resources and services availability, quick accessibility and scalability [4]. Further, CC saves on cost, saves on energy, is time-effective and is rapidly developing and empowering customers. These are the main influential factors that are leading to customers (individuals and corporates) being increasingly likely to adopt this technology [5]. Today's businesses are embarking on sweeping digital transformation (DX) initiatives to fundamentally retool business operations and rethink entire business models through the strategic use of digital technologies such as cloud services, mobile applications and data analytics. The broad adoption of cloud applications is helping support a surge in remote workers. The flexibility and scalability of cloud services and applications make these technologies a prerequisite for all modem DX strategies. This is forcing businesses around the world to embrace an essential cultural shift in the relationship between business and technology, one that empowers business units to leverage the self-service and on-demand nature of cloud services to transform the business with new levels of agility. According to 2020 Oracle and KPMG Cloud Threat Report [6], the symbiotic relationship between these DX strategies is directly correlated with cloud adoption, per the 88% of organizations who have attained a more mature level of digital transformation by utilizing cloud services-specifically, infrastructure services, which are seen as critical enablers of the 2020 economy. Furthermore, Oracle and 2020 KPMG Cloud Threat Report [6] revealed a shift in attitudes towards cloud security, with 75% of respondents viewing the public cloud as more secure than their own data centers. This data was developed through an online survey of 750 cybersecurity and IT professionals working across the United States, Europe and Asia. In essence, cloud adoption continue expand; digital transformation, cloud-first initiatives and a bullish level of confidence in the security of public clouds is driving an expanded use of cloud services.

In spite of the tangible advantages of cloud computing, but there are some issues that need to be addressed. Among them, the security concern is the first and has become a huge impediment to the development of cloud computing. Further, it has turned into one of the main brakes that slow down the rate of adoption of Cloud computing [7]. In actual fact, the well-known internet security corporate Symantec highlights in its 2019 Internet Security Threat Report [8] that security Cloud challenges merge on multiple fronts; from simple misconfiguration issues to vulnerabilities in hardware chips,

in 2018 we saw the wide range of security challenges that the cloud presents. Poorly secured cloud databases continued to be a flimsy point for organizations. In 2018, S3 buckets emerged as an Achilles heel for organizations, with more than 70 million records stolen or leaked as a consequence of bad configuration. This was on the heels of a spate of ransomware attacks against open databases such as MongoDB in 2017, which saw assailants erase their contents and demand payment with a view to restore them. Attackers didn't stop there also targeting container deployment systems such Kubernetes, serverless applications and other publicly exposed API services. There is a common theme across these incidents mediocre configuration. A more insidious threat to the cloud arisen in 2018 with the revelation of plural vulnerabilities in hardware chips. Meltdown and Spectre leverage vulnerabilities in a process known as speculative execution. Successful exploitation gives access to memory locations that are normally prohibited. This is peculiarly problematic for cloud services because while cloud instances have their own virtual processors, they share pools of memory, this means that a successful attack on a single physical system could result in data being leaked from multiple cloud instances. Meltdown and Spectre weren't isolated cases—several variants of these attacks were thereafter released into the public domain across the year. They were likewise followed up by similar chip-level vulnerabilities such as Speculative Store Bypass and Foreshadow, or L1 Terminal Fault. This is probably just the start, as researchers and attackers home in on vulnerabilities at the chip level, and point that there are arduous times ahead for the cloud [8]. Furthermore, 2020 Oracle and KPMG Cloud Threat Report outlines two major security issues [6]:

- Cyber fraud takes center stage: The threat landscape is evolving, with tried and true phishing attacks leading to an increase in cyber business fraud and compromised privileged cloud credentials.
- Misconfigured cloud services are prevalent, problematic and the top cloud security priority**:** A cloud security visibility gap has made hardening the configuration of cloud services a systemic challenge.

Finally, on the basis of Netskope Cloud and Threat Report - July 2021 [9], the risk of data exposure is more apparent than ever due to the growth of enterprise cloud application adoption and shadow IT, increased personal app usage, and third-party app plug-ins. Additionally, departing employees present disproportionately significant cloud security risks, meaning that if a "Great Resignation" is happening, it might make enterprises even more vulnerable to cloud-borne threats. Highlights included in the Netskope report are:

- Departing employees upload 3X more data to personal apps in their final month of employment.
- 97% of Google Workspace users have authorized at least one third-party app.
- Cloud-delivered malware has increased to an all-time high of 68%.

The security reports cited formerly indicate that providing privacy and security to protect cloud assets and resources still a very challenging issue. The hackers intensively target the cloud systems for exploiting its involved vulnerabilities [10]. Any unauthorized access by an intruder to the cloud is commonly known as an intrusion, while the intrusion detection is the process of monitoring and auditing the events that occur in the systems of the computers or the networks. Detecting and preventing network intrusions in the cloud environment are still from the primary security worries among researchers [4]. In recent years, since the unknown attacks are continuously increased, the conventional network protection tools, such as firewalls, access control or encryption, fail to defend computer networks included cloud networks against the novel attacks. Consequently, the efforts presently are dedicated on establishing the more complex systems or network architectures, e.g., multidimensional context-aware, quality-aware service access system, cyber physical systems, emotion-aware cognitive system, social network architecture and NIDS (network Intrusion Detection System) etc. Among these security applications, the NIDS increasingly attracts attentions [11]. NIDS analyses the network and checks for any malevolent activity. If malevolent activity turns up, it warns the network administrator at once. It sometimes even block IP address of the user from accessing the network who is attempting to intrude [12]. Thereby, NIDS preserve the confidentiality, integrity and availability (CIA) of the networks and information systems in the cloud environment, since it plays pivotal roles in the security provisioning against the intruders [4]. In a traditional network, the nodes are fixed, whilst in the cloud, the nodes are likely to shift from one physical machine to others. In the scene of cloud computing, traditional intrusion detection methods lack practicality. The intrusion detection systems deployed in a traditional network cannot be applied to such systems owing to the dynamic nature of logical resources [7]. Thus, efficient intrusions detection in Cloud environments requires embracing of new intelligent techniques such as Machine Learning (ML) techniques [13].

One of the principal ML techniques that has successfully used in addressing complex practical challenges is DNN. DNNs have the ability to solve several problems confronted by the other current techniques used in intrusion detection [14]. There are five advantages of intrusion detection based on DNN [15, 16]:

- DNN has the ability to process data from a number of sources in a non-linear fashion. This is very important especially when coordinated attack by multiple attackers is conducted against the network.
- DNN provides elasticity in intrusion detection process, where DNN has the ability to analyze and ensure that data right or partially right. Likewise, DNN is capable of performing analysis on data in nonlinear fashion.
- High capability of generalization.
- Remarkable classification performance.
- DNN [17] can automatically reduce the complexity of network traffic by finding the data correlation without human intervention. They also contribute to reducing the rate of type positives and increasing the detection rate in anomaly detection systems.

In this paper, we focus on the anomaly detection, because theoretically, it is capable of detecting both known and new unseen attacks, and under the current complicated Cloud network environment, the anomaly detection is much more required and has a better application foreground [18].

In this work, we present a new powerful machine learning based intrusion detection system for cloud environments, developed with the purpose to reduce impact of network attacks (known attacks, and unknown attacks), while ensuring higher detection rate, lower false positive rate, higher accuracy and higher precision with an affordable

computational cost. We suggest a smart approach to construct automatically a network intrusion detection system (NIDS) based on Deep Neural Network (DNN), by using a novel hybrid optimization framework termed "ISAGASAA" that combines our self-adaptive heuristic search algorithm called "Improved Self-Adaptive Genetic Algorithm (ISAGA)" and Simulated Annealing Algorithm (SAA). SAA is incorporated to ISAGA with the aim to optimize its heuristic search. DNN has been widely studied in machine learning research field and amply used for practical applications in image processing, computer vision and speech recognition, etc. [19]. DNN is adopted in this study due to its appealing features in terms of intrusion detection mentioned formerly. ISAGA is our variant of standard Genetic Algorithm (GA), which is developed based on GA, improved through an Adaptive Mutation Algorithm (AMA) [20] and optimization strategies. AMA allows to automatically adjust the mutation rate should be applied for any given individual from the population of ISAGA, in order to augment the chance of preserving individuals that are performing well versus the optimization problem in hand and reduce the chance of preserving individuals that don't perform well. That tuning or adjustment of mutation rate takes place while ISAGA is running, hopefully resulting in the best parameters being used at any specific time during execution. It is this continuous adaptive adjustment of ISAGA parameters that will often result in its performance improvement. Further, ISAGA is optimized through optimization strategies, namely Parallel Processing and Fitness Value Hashing, which reduce execution time, convergence time and save processing power. As the fitness function is typically the most computationally expensive component, and it is often going to be the bottleneck of GA, this makes it an ideal candidate for multi-core optimization (Parallel Processing). By using multiple cores, it is possible to compute the fitness of numerous individuals simultaneously. Besides, Fitness Value Hashing is another strategy that can reduce the amount of time spent computing fitness values by storing previously calculated fitness values in a hash table. Thereby, when a previously visited solution (chromosome) is revisited, its fitness value can be retrieved from the hash table, avoiding the need to recalculate it. Our approach consists of using ISAGASAA framework with the goal of searching the optimal or near-optimal combination of most relevant values of the parameters included in construction of DNN based IDS or impacting its performance, like feature selection, data normalization, architecture of DNN, activation function, learning rate and momentum term, which ensure high detection rate, high accuracy and low false alarm rate. In addition, the ANIDS resulted named "MLANIDS" (Machine Learning based Anomaly Network Intrusion Detection System) is designed to be deployed in both front-end and back-end of the cloud. Consequently, that helps to detect attacks from external network of the cloud and also internal attacks either in internal physical network or virtual network within hypervisors.

The rest of this paper is organized as follows: Section 2 gives the literature surrounding network intrusion detection systems (NIDSs). Section 3 explains the background related to this study, such Simulated Annealing, Adaptive Mutation Algorithm and optimization strategies of GA as Parallel processing and Fitness value hashing. Section 4 presents the proposed system in detail, describes its work, explains the role of Simulated Annealing Algorithm in this system and provides the framework of our model. Section 5 introduces positions of the proposed system in a Cloud Network. Detailed description of Kyoto version 2015 and CIDDS-001 datasets, experimental results obtained based on those datasets and analysis are provided in section 6. Finally, section 7 ends with the conclusions and Future work.

2. Literature Review

Kim and Gofman [21] have compared the efficacies of shallow network to deep neural network for network intrusion detection based on the NSL-KDD dataset. The shallow network used comprises an input layer, a single hidden layer and an output layer. Further, it uses the scaled conjugate gradient descent (SCG) backpropagation algorithm as learning algorithm and tanh function as activation function. While, the deep neural network employed includes the same aspects of the shallow neural network except the number of hidden layers. Instead of a single hidden layer, the deep network has two hidden layers. The experiments performed using MATLAB version 2016b and NSL-KDD illustrate the superior performance of shallow network, which achieves a 98.50% detection rate of malicious traffic and a 1.40% false positive rate. In opposite, according to experiments carried out deep neural network have classified all observations as malicious packets, thus true negative and false negative rates for deep neural network (DNN) were both 0. In our point of view, the poor outcomes obtained for DNN are due to two causes. Firstly, the authors have not taken in consideration in their study important factors that affect the performance of both shallow and deep neural networks, namely feature selection and data normalization technique. In fact, Woo et al. [22] have enhanced the performance of intrusion detection systems based on DNN through the use of feature selection and layer configuration, as result, the IDS based DNN built have reached an average accuracy of 98.5%. Secondly, the method used to determinate the number of nodes in hidden layers was not indicated. In effect, the method followed affects considerably the performing of shallow and deep neural networks. As stated in the paper [23], the random selection of a number of hidden neurons might cause either overfitting or underfitting problems. To overcome these issues, authors of the work [23] present several solutions and propose their proper approach.

In order to enhance the performance of intrusion detection systems based on deep neural network, Woo et al. [22] have proposed to use feature selection and layer configuration. The goal of using feature selection is to eliminate features that are less relevant between features and avoid over-fitting, thereby reducing learning time and improving accuracy of IDSs. For feature selection, the authors have adopted Pearson Correlation method. Concerning layer configuration, three cases were investigated; case 1, case 2 and case 3 are composed of 5, 4 and 3 layers respectively. For experimentation, Python language, Keras API and NSL-KDD were used. Experimental results obtained demonstrate that Pearson Correlation method allows removing a feature named 'num_outbound_cmds', resulting in reduction of learning time and increasing accuracy. In addition, it is found that case 2 is the best layer configuration that allow avoiding overfitting and enhancing the performing. Hence, applying Pearson Correlation feature selection jointly with architecture of 4 layers for DNN raise clearly the performance of IDS based DNN, leading it to reach an average accuracy of 98.5%.

To classify all the attacks properly and proficiently, IDS is required for securing the Cloud data. For getting a better classification accuracy, a proper training of IDS is significant. The presence of irrelevant features in training data set increases training time, deteriorate classification accuracy as well as increases memory representation. To overcome the aforementioned issues, Ghosh et al. [12] have proposed and implemented a novel feature selection method CS-PSO (Cuckoo search (CS) - Particle swarm optimization (PSO)) for producing an efficient IDS that classify attacks accurately and rapidly in cloud environment. The proposed IDS model is comprised of two components; the first is CS-PSO algorithm, which is prepared by combining the CS and PSO meta-heuristic methods, and it is employed for feature selection. While the second is Logistic Regression (LR) classifier that classifies samples extracted from NSL-KDD dataset as normal or attack. In the experiments conducted by the authors based on NSL-KDD dataset, the CS-PSO algorithm proves its efficacy by solving the trade compromisation between exploitation and exploration. In the proposed model, the authors have leveraged the efficiency of PSO in finding the best value in a local area and is a very good example of exploitation concept. To take care of the exploration process the CS approach does its job through the use of Levy Flight technique. It jumps into random positions to get better global optimized value. The conjunction of these two nature inspired techniques have brought some fruitful results with high accuracy (79.32%) reached by Logistic Regression classifier, and thus producing efficient IDS in Cloud Environment. However, detection rate of 65.10% yielded by the proposed system need to be enhanced, and the false positive rate of 2.87% achieved is relatively high.

Zhang et al. [24] have built an intrusion detection model based mainly on Elman neural network, which is optimized by using Mind Evolutionary Algorithm (MEA). Further, they have employed Genetic Algorithm (GA) in order to reduce the dimension of KDDCUP99_10% dataset used for training and testing the MEA-Elman network model. The proposed model goes through two stages. The first stage consists firstly of construction of an Elman neural network composed of four layers: input layer that contains 41 nodes corresponding to the 41 features of KDDCUP99_10% dataset, ten hidden layers, one context layer and one output layer of five nodes (Normal, Probe, DOS, U2R and R2L). Thereafter, MEA is used to optimize the initial weights and thresholds of this network. Evaluation of the MEA-Elman network model built in the first stage highlights that it has a high accuracy of Normal, Probe and DOS classification of more than 90%, but the accuracy of the U2R and R2L classification is too low. To address this issue, the GA was used in the second stage as a data processing algorithm with the goal of reducing dimension of the dataset used by the proposed IDS model based MEA-Elman network. That may help to improve the accuracy of U2R and R2L classification. At the end of GA process, the MEA-Elman model is rebuilt using the subset of relevant features among the 41 features of KDDCUP99_10% dataset. Indeed, the experimental results show that the detection accuracy of U2R and R2L has been greatly improved by using the GA algorithm. However, detection accuracy of Normal, Probe and DOS has decreased slightly in comparison the obtained results following the evaluation of the system before application of GA, but it still keeps around 90%. The average accuracy of the proposed system is 77.82%. Thus, it need to be improved.

In the paper [4], an anomaly-based network intrusion detection system (NIDS) have been developed to detect different types of attacks (R2L, U2R, Dos and Probe) in the cloud environment. In the proposed model, Support Vector Machine (SVM) was employed as the classifier of the network connections. To reinforce the efficiency of intrusion detection process, decrease its detection time and build a lightweight IDS, the binary-based Particle Swarm Optimization (BPSO) was adopted to reduce the dimensionality of the network connection features by selecting the most relevant/optimal features subset from the incoming network data flow. Further, as the values of the SVM control parameters, namely the penalty parameter (C) and the RBF kernel parameter (σ) have a great impact on its performance; SPSO (standard-based binary-based Particle Swarm Optimization) searching algorithm was chosen to search the best values of these parameters in order to optimize the SVM performance. The proposed NIDS was trained and tested on the benchmark NSL-KDD dataset, and the evaluation results proved that it outperforms other related IDSs with a higher classification accuracy of 99.10%, detection rate of 99.08% and a lower FPR of 0.87%. It is required to evaluate the proposed IDS based on a recent IDS dataset that contains novel attacks, in order to prove its efficacy in recognizing the normal behaviors and detecting the attacks with high detection accuracy and low rates of false alarms.

Ghanshala et al. have [25] proposed a light weighted and adaptable intrusion detection approach named as Behavior-based Network Intrusion Detection (BNID) to detect attacks at network-layer in Cloud Environment. A security framework was developed by the authors for deployment of BNID at Cloud Network Node (CNN) of the cloud network to defend the cloud machines against network attacks. BNID captures the behavior of TVM (Tenant Virtual Machines) communication, analyzes it and detects the malicious network pattern at the network-layer. Thus, the need of deploying security tool at each TVM is eliminated. In addition, it is adaptable to learn the behavior of newly generated traffic patterns. For traffic behavior analysis, BNID uses statistical learning techniques with feature selection. In fact, firstly, network traffic is captured and important features are extracted using fusion of two popular feature selection methods i.e. Recursive Feature Elimination (RFE) and Chi Square. This removes the less relevant features and accelerate the processing of data, which improves the classifiers' performance. Subsequently, the processed traffic resulted from the former step (capture & feature selection), is analyzed using Random Forest (RF) classifier for detecting intrusive activities. Like other supervised learning techniques, RF works in two stages; learning stage and detection stage. BNID compromises four detection modules namely (1) Packet Capture (PacCap()), (2) Packet Pre-processor (PacPre()), (3) Detection Engine (DetEng()) and (4) Alert and Log Generator (AltGen()). PacCap() generates the collection of packet capture information logs, and It also loads the known attack dataset that serves as training dataset providing the behavior statistics of different types of attacks. Packet Pre-Processor (PacPre()) module pre-processes the packets and selects relevant features to construct feature vectors. DetEng() is responsible for learning and detecting the attack behavior. AltGen() generates the alerts once attack signal is

received from DetEng() module. A report is generated by AltGen() and sent to cloud administrator for further analysis. Experimental results based on ITOC dataset demonstrates that BNIDS yields better performance than Naive Bayes (NB) and Decision Tree C 4.5 (DT) techniques in terms of intrusion detection. It achieves an accuracy of 99.182% with 1.517% FPR. However, the proposed system can classify only attacks whose behavior is pre-identified and the variants of such attacks, hence, it is not able to detect unseen attacks.

Shyla and Sujatha [26] have suggested a novel IDS established on a combination of a leader-based k-means clustering algorithm (LKM), fuzzy logic system (FLS) and grey wolf optimization (GWO). The overall process of the proposed model is split into two stages, namely, training and testing. The dataset utilized in that work is NSL- KDD CUP 99 dataset. In training phase, the data extracted from the training dataset are preprocessed. Then, to reduce the complexity, the preprocessed data are clustered using LKM. After the clustering process, each obtained cluster is given to an optimal fuzzy logic system (OFLS). The number of clusters and the OFLSs are identical. In this, the FLS rules are optimally selected using a bio-inspired algorithm, namely, GWO. GWO is employed to reduce the time complexity and increase the detection accuracy. In testing stage, at first, the incoming data are preprocessed. After preprocessing, the clustering process is applied on the preprocessed data. Subsequently, the data are afforded to the corresponding cluster based OFLS. The trained OFLS structure tests the data. Finally, the score value is obtained. If the obtained score value is above the threshold T_h, it means the data are intruded; otherwise the data are normal. The experimental results applying NSL-KDD CUP 1999 dataset and Cloud Sim tools demonstrate the superiority of the proposed approach in comparison to several existing methods. In fact, it achieves precision of 96.54%, recall of 89.26% and F-measure of 91. 83%. Nevertheless, more enhancements are required to improve achievement of proposed system, especially recall and F-score performances.

Cloud IDSs often suffer from poor detection accuracy due to coordinated attacks such as a DDoS. Hence, various research on distributed IDSs have been proposed to detect DDoS. However, the limitations of these works the lack of technique to determine an appropriate period to share attack information among nodes in the distributed IDS. Therefore, the paper [27] proposes a Distributed Cloud Intrusion Detection Scheme (D-CIDS) that uses a Binary Segmentation Change Point Detection Algorithm to address the appropriate period to send attack information to nodes in distributed IDS and using parallel Stochastic Gradient Descent with Support Vector Machine (SGD-SVM) to achieve the distributed detection. D-CIDS is comprised of five components namely: the feature selection component, the distributed classifier training component, the distributed attack detection component, the Destination-IP monitoring component and the aggregation component. The feature selection component uses a hybrid Ant Colony Optimization (ACO) and Correlation-based Feature Selection (CFS) for selection of 16 relevant features among 41 features of NSL-KDD dataset, whilst the distributed classifier training component trains each node in the distributed IDS using SGD-SVM to create a local reference model for anomaly detection. The detection component detects attacks based on the reference model created during training and sends the LRM to the aggregating unit (master node) when a threshold is exceeded, while the Destination-IP monitoring component utilizes binary segmentation change point algorithm to monitor the destination-IP count from same host to determine the appropriate time to share attack information among the nodes in the distributed IDS. Whereas, the aggregation component aggregates the LRM from each node to create a Global Reference Model (GRM), which allows classifying network traffic as normal or intrusion. The proposed system was implemented using MLlib Apache Spark distributed machine learning framework. The performance of D-CIDS was evaluated using NSL-KDD intrusion detection dataset, and performance comparison made with related works shows that D-CIDS achieved superior performance in terms of accuracy (99.6%), detection rate (99.7%) and false positive rate (0.03%). Recent IDS datasets that contain new DDoS attacks should be used to assess the effectiness of the proposed IDS.

Rabbani et al. [28] have proposed a new hybrid machine learning approach for malicious behaviour detection and recognition in Cloud Computing. They applied a Particle Swarm Optimization-based Probabilistic Neural Network (PSO-PNN) for the detection and recognition process. The proposed system includes two main modules, Data Preprocessing and Recognition. The former is designed for data preparation and is used to extract informative features from the UNSW-NB15 dataset for learning and modelling purposes. The pre-processing involves four steps: feature creation, reduction with the help of PCA (Principal Component Analysis), conversion of non-quantitative features into numeric ones and feature normalization by means of Min-Max method. The latter module is the core of the proposed technique for behavioural recognition; it includes two phases, training and prediction. In training phase, recognition module models the users' activities based on Probabilistic Neural Network (PNN) optimized through Particle Swarm Optimization (PSO). Statistically, the performance of a PNN is largely influenced by the spread parameter (α), which plays an important role in increasing the classification rate in PNN systems. In this hybrid technique, the determination of σ is obtained by PSO. It is initialized with a swarm value of σ, and the optimization value is computed for a specific pattern taken from the training dataset. Subsequently, the same procedure can be applied for each pattern in the training dataset to finalize a global σ; therefore, this optimization model makes the structure of PNN a self-adaptive network. In prediction/detection phase, the recognition model built in training phase is used to predict whether a particular user is normal or malicious. Evaluation results obtained based on UNSW-NB15 dataset demonstrate the superiority of the PSO-PNN technique in terms of Recall/Detection Rate (96.4%), Precision (96.4%) and F-measure (97.5%) when compared with new state-of-the-art network-based intrusion detection techniques. However, the proposed method have limited ability to properly distinguish between Back door and Analysis attacks. Moreover, the False Positive Rate (FPR) attained (3.6%) is relatively high, it need to be reduced.

Neha et al. [29] have presented a Salp Swarm Optimization based Feed Forward Neural Network (SSO-FFNN) to build an intrusion detection methodology for computer networks. SSO was introduced to optimize the hyper parameters of FFNN to achieve the major goals of IDS, which are yielding high detection rate, less false alarm rate and time complexity. The experimentations were carried out using WEKA tool and the standard NSL-KDD cup intrusion dataset and the

implementation of SSO-FFNN was evaluated with regards to classification accuracy, detection rate and false alarm rate. Experimental outcomes indicate that the proposed system shows better results than several existing machine learning based Intrusion detection techniques in terms of performance metrics such as detection rate and false alarm rate, and that it achieves 98.63% accuracy. However, for the weighted fitness function adopted for evaluating the performance of the proposed system, that is composed of Detection Rate (DR) and False Positive Rate (FPR) metrics, the method followed for calculation the weights of DR and FPR did not provided by the authors.

Krishnaveni et al. [30] have proposed an anomaly intrusion detection system for cloud computing using machine learning algorithm, namely Support Vector Machine (SVM) classifier and a filter-based feature selection algorithm such as Information Gain Ratio (IGR) method. In this approach, features has been ordered using an impartial measure: Information Gain Ratio. This is done by calculating the IGR which requires the initial entropy values. The proposed algorithm (IGR) is used to pick up the significant feature set from the NSL-KDD dataset. The condensed feature NSL-KDD dataset is then utilized for training and testing the detection model relied on the SVM classifier. During the experiments carried out by the authors, a comparison of accuracy of different kernel functions of SVM with feature selection was performed. The kernel function used was linear, Gaussian, RBF and polynomial. The results show that the RBF (Radial Basis Function) kernel function with optimized features gives the highest accuracy. Moreover, the proposed system was then checked for its accuracy and its response time by using the different classifiers, i.e., RBF SVM, logistic regression and K nearest neighbor. The outcomes obtained demonstrate that RBF SVM was achieved higher performances than the other methods; higher accuracy (96.34%) and lower response time (4.90s). Nevertheless, others prominent performance evaluation metrics was not employed to assess the proposed system like False Positive Rate and Detection Rate. In addition, recent IDS datasets need to be used to evaluate further efficiency of the proposed IDS.

Abirami et al. [31] have developed an anomaly and feature based IDS by means of a feature selection algorithm (FSA) and an ensemble learning algorithm or stacking classifier. They have employed Principal Components Analysis (PCA) as FSA for reducing the dimensionality of feature space; hence, it identifies the important features and eliminates irrelevant ones. Moreover, they have combined Random Forest, Linear Support Vector Machine (SVM) and Naive Bayes methods by using Logistic Regression as meta-classifier, as result, building a stacking classifier. Experimental outcomes obtained based on UNSW-NB15 data set have proved that the proposed IDS have achieved highest accuracy of 95%, in comparison with Random Forest, Linear SVM and Naive Bayes classifiers used individually.

Thilagam and Arunaproposes [32] have proposed a novel IDS for cloud computing, which is based on an innovative and optimized custom RC-NN (Recurrent Convolutional Neural Network). RN-NN network model is created by connecting the layers of LSTM and CNN, furthermore, it is optimized with the help of Ant Lion optimization (ALO) algorithm that is utilized within the proposed network layers to minimize the error rate thus improving accuracy. Experimental findings obtained by the authors prove the superiority of the developed model in terms to intrusion detection, compared to other existing classifiers like LSTM, CNN and LSTM with CNN. The accuracy is identified as 0.9401 for DARPA dataset and 0.9428 for CSE-CIC-IDS2018 dataset. Nevertheless, DARPA dataset is an old dataset, which does not reflect the recent trend of cyber-attacks. Hence, new intrusion datasets such as UNSW-NB15 (2015) / CICIDS18/ CIDDS-001 are required to assess the effectiveness of the suggested model.

In the paper [33], a novel Intrusion detection system (IDS) is conceived to fix the security concerns that unfavorably affect sustainable development of cloud and to improve the defence of cloud from malevolent attacks. The IDS is modeled using proposed Feedback Deer Hunting Optimization (FDHO)-based Deep Residual network to uncover network intrusions. The major processes included in the suggested approach is pre-processing, feature extraction, binary classification, and attack detection. The data is pre-processed by exponential kernel and the preprocessed result is allowed to feature selection module. In feature selection step, the rough set and entropy based model is utilized to pick out the optimal and best features. Based on the selected features, the process of binary classification is performed using SVM. With SVM classifier, the data can be classified into normal or attack by considering the features. When it is identified that the data is attack, finally the process of attack detection is done using Deep Residual Network in such a way that the training procedure of detection approach is carried out using proposed FDHO algorithm. The developed FDHO algorithm is the integration of Feedback Artificial Tree (FAT) with Deer Hunting Optimization (DHOA). The proposed model is assessed sing BoT-IoT dataset and KDD cup-99 dataset. However, benchmarking dataset KDD CUP' 99 is outdated, and it is not included new type of network attacks, which make it unsuitable for evaluating Anomaly based Network Intrusion Detection Systems.

3. Related Background

This section provides the necessary background to understand the problem in hand. First subsection shed the light on a novel Improved Self-Adaptive Genetic Algorithm (ISAGA) that is combined in this work with Simulated Annealing Algorithm (SAA), to build automatically an innovative IDS based on Deep Neural Network. As mentioned formerly, ISAGA is our variant of standard Genetic Algorithm (GA), which is developed based on a GA, improved through an Adaptive Mutation Algorithm (AMA) and optimization strategies incorporated to GA, namely Parallel Processing and Fitness Value Hashing. Whereas, the second subsection exhibits briefly Simulated Annealing Algorithm. We did not offer the fundaments and concepts of Deep neural network [15, 16, 17, 21, 23, 24] and Genetic Algorithm [20, 26, 34, 35], since they are exhaustively presented in the literature.

3.1 Improved self-adaptive genetic algorithm

Improved Self-Adaptive Genetic Algorithm (ISAGA) is our variant of standard Genetic Algorithm (GA), which is created based on GA, enhanced via an Adaptive Mutation Algorithm (AMA) and optimization techniques applied to GA, that is to say Parallel Processing and Fitness Value Hashing.

3.1.1 Adaptive Genetic Algorithms : Adaptive Mutation Algorithm

Adaptive Genetic Algorithms (AGA) [20] are a popular subset of genetic algorithms, which can provide significant performance improvements over standard implementations

when utilized in the suitable circumstances. A key factor that determines how well a genetic algorithm (GA) will perform is the manner in which its parameters are configured. Thus, finding the right values for the mutation rate and crossover rate plays in substantial role when building an efficient and effective GA. Typically, configuring the parameters will require some trial and error, together with some intuition, before eventually attaining a satisfactory configuration. AGA are useful because they can help in the tuning of these parameters automatically by adjusting them based on the state of the algorithm. These parameter adjustments take place while GA is running, hopefully resulting in the best parameters being used at any specific time during execution. It is this continuous adaptive adjustment of GA parameters that will often result in its performance improvement. AGA used in this work uses information such as the average population fitness and the population's current best fitness to calculate and update its parameters in a way that best suits its present state. For example, by comparing any specific individual to the current fittest individual in the population, it's possible to gauge how well that individual is performing in relation to the current best. Typically, we want to augment the chance of preserving individuals that are performing well and reduce the chance of preserving individuals that don't perform well. One way we can do this is by allowing the algorithm to adaptively update the mutation rate. We can determine if the algorithm has started to converge by calculating the difference between the current best fitness and the average population fitness. When the average population fitness is close to the current best fitness, we know the population has started to converge around a small area of the search space. When calculating what the mutation rate should be for any given individual, two of the most important factors/characteristics to consider are how well the current individual is performing and how well the entire population is performing as a whole. The algorithm we had used in this work to assess these two characteristics and update the mutation rate is called **Adaptive Mutation Algorithm (AMA)**, and it is defined by equations (1) and (2).

F_i : Is the fitness value (score) of the current individual identified by the index i.

F_{max} : Is the best fitness from the population.

F_{avg} : Is the average population fitness.

m: Is the mutation rate that was set during initialization of GA.

P_m: Is the new mutation rate that should be applied for the current individual.

$$P_m = (F\max - Fi) / (F\max - Favg), \quad Fi > Favg \quad (1)$$

$$P_m = m, \quad Fi \pounds Favg \quad (2)$$

As shown by the equation 1, when the individual's fitness (F_i) is higher than the population's average fitness (F_{avg}), firstly, we calculate the difference between F_{max} and F_i. Afterwards, we compute the difference between F_{max} and F_{avg} and perform the division of the two resulted values. At last, we use the quotient of previous division to scale the mutation rate (m) that was set during initialization. Otherwise, as indicated by equation 2, if the individual's fitness is the same or less than the population's average fitness, we simply use the mutation rate as set during initialization. Adaptive genetic algorithm can be employed to adjust more than just the mutation rate however. Similar technique can be applied to adjust other parameters of the genetic algorithm like the crossover rate to get further improvements as needed.

3.1.2 Optimization Strategies for Genetic Algorithm

With the fitness function, typically being the most processing demanding component of genetic algorithm (GA), it makes sense to focus on improvement of the fitness function to see the best return in performance. In this subsection, we will explore two optimization strategies that are used in this work to improve performance of GA by optimizing the fitness function, namely Parallel Processing and Fitness Value Hashing.

(a) Parallel Processing

One of the easiest approaches to achieve a performance enhancement of GA is by optimizing the fitness function. The fitness function is typically the most computationally expensive component; and it is often going to be the bottleneck of GA. This makes it an ideal candidate for multi-core optimization. By using multiple cores, it is possible to compute the fitness of numerous individuals simultaneously, which makes a tremendous difference when there are often hundreds of individuals to evaluate per population. Java 8 provides some very useful libraries that make supporting parallel processing in our GA much easier. Using ***Java's IntStream***, we can implement parallel processing in our fitness function without worrying about the fine details of parallel processing (such as the number of cores we need to support); it will instead create an optimal number of threads depending on the number of cores available in our multi-core system. Hence, by using parallel processing, fitness function will be able to run across multiple cores of the computer, consequently, it is possible to considerably reduce the amount of time the GA spends evaluating individuals and, so reduce the overall time of execution of GA, and accelerate convergence process [20].

(b) Fitness Value Hashing

Fitness Value Hashing is another strategy that can reduce the amount of time spent computing fitness values by storing previously calculated fitness values in a hash table [20]. During running of GA, solutions found previously will occasionally be revisited due to the random mutations and recombinations of individuals. This occasional revisiting of solutions becomes more common as GA converges and begins to find solutions in an increasingly smaller area of the search space. Each time a solution is revisited its fitness value needs to be recalculated, wasting processing power on recurrent, duplicate computations. Luckily, this can be easily fixed by storing fitness values in a hash table after they have been computed. When a previously visited solution is revisited, its fitness value can be retrieved from the hash table, avoiding the need to recalculate it.

3.2 Simulated annealing algorithm

Simulated Annealing is motivated by an analogy to the statistical mechanics of annealing in solids [36], which coerces a solid (i.e., in a poor, unordered state) into a low energy thermodynamic equilibrium (i.e., a highly ordered defect-free state) such as a crystal lattice. Annealing is referred to as tempering certain alloys of metal, glass, or crystal by heating above its melting point, holding its temperature, and then cooling it very slowly until it solidifies into a perfect crystalline structure. This physical/chemical process aims to get the ground state of matter, which is the minimal energy of the solid state [37], as result, it produces

high-quality materials [38]. The idea of SA comes from a paper published by Metropolis et al. in 1953 [39]. The algorithm in this paper simulated the cooling of material in a heat bath. This process as mentioned previously is known as annealing. If you heat a solid past melting point and then cool it, the structural properties of the solid depend on the rate of cooling. If the liquid is cooled leisurely enough, large crystals will be formed and thus an ideal crystalline structure is obtained. However, if the liquid is cooled rapidly (quenched) the crystals will contain imperfections. Metropolis algorithm simulated the material as a system of particles, and the cooling process by gradually lowering the temperature of the system until it converges to a steady, frozen state named the ground state [36]. The simulation in the *Metropolis algorithm* computes the new energy of the system in a new state gotten by a random perturbation. If the energy has decreased then the system moves to this state. If the energy has increased then the new state is accepted according to a certain probability. A certain number of iterations are carried out at each temperature and then the temperature is lowered. This is repeated until the system freezes into a steady state named ground state.

In 1982, Kirkpatrick et al. [40] took the idea of the Metropolis Algorithm and developed a variant of that algorithm called *Simulated Annealing Algorithm (SAA)* to solve optimisation problems. The idea is to use simulated annealing to search for feasible solutions and converge to an optimal solution [36]. Kirkpatrick et al. have successfully solved Vehicle Routing Optimization, TSP problem, Workshop Scheduling Optimization problems, etc [41]. Ever since, SAA has been used in a great number of problems, including NP-hardy combinatorial problems. In combinatorial optimization problems, the purpose is developing efficient techniques for finding minimum (or maximum) values of a function with many degrees of freedom and many local minima. States in thermodynamic usage correspond to solutions in the combinatorial optimization problem. Energy in thermodynamics is the cost function in simulated annealing. The ground state, change of state, and temperature in thermodynamics translate to the optimal solution, a neighbouring solution, and the control parameter in simulated annealing, respectively [42]. The table 1 extracted from the paper [36], shows how physical annealing can be mapped to simulated annealing. Using these mappings any combinatorial optimisation problem can be converted into a simulated annealing algorithm [36, 40].

Some application of SAA are:

Basic Problems:

- Traveling salesman
- Graph partitioning
- Matching problems
- Graph coloring
- Scheduling.

Engineering:

- VLSI design.
    1. Placement
    2. Routing
    3. Array logic minimization
    4. Layout
- Facilities layout
- Image processing
- Code design in information theory

Table 1. Relationship between physical annealing and simulated annealing

Thermodynamic Simulation	Combinatorial Optimisation
Metal (System of particles)	Problem
System states	Feasible solutions
Energy	Cost function
Change of state	Neighbouting solutions
Temperature	Control parameter
Careful annealing	Simulated annealing
Frozen state or Ground state (A completely ordered crystalline annealing)	Optimal solution for the problem

4. The Proposed System

This section describes in detail our new proposed IDS and gives the model of that IDS.

4.1 Approach of our proposed system

In this work, we propose a clever approach to build automatically a network intrusion detection system (NIDS) based on Deep Neural Network (DNN), by using a novel hybrid optimization framework termed “ISAGASAA” that combines our self-adaptive heuristic search algorithm called “Improved Self-Adaptive Genetic Algorithm (ISAGA)” and Simulated Annealing Algorithm (SAA). SAA is incorporated to ISAGA with the purpose to optimize its heuristic search. ISAGA is our variant of standard Genetic Algorithm (GA), which is created based on GA, enhanced through an Adaptive Mutation Algorithm (AMA) (subsection 3.1.1) and optimization strategies (subsection 3.1.2). Our DNN is a Back Propagation Neural network (BPNN) with one input layer, two hidden layers and one output layer. The number of nodes in the input layer matches to the number of attributes/features in the vector of connection instance pulled out from IDS dataset and introduced to DNN, whereas the number of nodes in each hidden layer will be generated by ISAGA. While, the output layer comprises one node, which provides a value of 1 in case of classification of input pattern by DNN as normal/legitimate traffic; or else, it gives a value of 0 to point out an intrusion. Our approach consists principally of **four phases**. In two first phases, we have studied heavily numerous works related to intrusion detection systems based on BPNN and DNN.

The first phase was concentrated on the determination of the most pertinent parameters used to construct that type of classifier or that affect its performance. As shown by table 2, at the end of our study, we have deduced that the most significant parameters are [18]:

- The number of selected features/attributes that corresponds to the number of nodes in the input layer.
- Normalization of data.
- Architecture of Neural Network, specifically the number of nodes in the hidden layer(s).
- Activation function or transfer function.
- Learning rate.
- Momentum term.

The second phase consists of comparison of studied works with the view to select for each parameter mentioned

foregoing, between two and four relevant values, which have yielded the best outcomes in terms of intrusion detection.

Table 2. List of parameters influencing the performance of a BPNN or a DNN based IDS and their different values

Parameters	Different values
Number of attributes	12 attributes NSL-KDD [20,43]
	10 attributes CIDDS-001 [44]
	14 attributes Kyoto 2006+ [45,46]
	27 attributes ITOS dataset [47]
Normalization	Min-Max normalization or Mean range [0,1] [48]
	Statistical normalization or Z-score [49]
Activation function	Hyperbolic tangent [50]
	Sigmoid [51]

In this study, ISAGA will generate randomly the number of nodes of both hidden layers of DNN and the values of Learning rate and Momentum term. With the help of genetic operations such selection, elitism, crossover and self-adaptive mutation, ISAGA algorithm is capable to found the optimal values of those parameters.

For training and evaluating of our proposed system, we have chosen primarily Kyoto 2006+ University Benchmark Dataset version 2015, which is one of the most recently used intrusion detection dataset in the IDS domain [46]. This dataset was developed by [52] to solve the problems of underperformance of machine learning based IDS model trained and tested on KDD or NSL-KDD and any other related old datasets heavily criticized for not reflecting the current trend of network situation and sophistication of ever evolving cyber-attacks [53]. Kyoto dataset is a multivariate attributes dataset that consists of 24 statistical features extracted from captured data including 14 conventional features that were derived from KDD Cup 99 benchmark dataset, and 10 additional features added for analysis and further investigation. This study uses Kyoto subset of **December 31, 2015** and has used the first 14 features (conventional features) that are suitable for network-based IDSs [54], and the label which indicates whether the record is an attack or normal, and excluded the features that can be used to investigate what kinds of attacks happened on computer networks [53]. Thereby, *the number of inputs* in our DNN is fixed at 14 inputs, which corresponds to the number of features selected in [53] from Kyoto 2006+ dataset. In addition, we have tested our approach on CIDD-001 dataset as outlined in subsection 6.3. Accordingly, we have also build another DNN with 10 inputs while using CIDDS-001 dataset.

The third phase: For successful use of ISAGA, two key elements must be well defined; *the representation/encoding of chromosomes* and *the Fitness Function.*

- **Chromosome encoding/representation:** In our study, we have adopted the binary representation for chromosomes. Each chromosome is a possible combination of values of the relevant parameters referred beforehand, that will be utilized to build an instance of IDS based DNN. Each parameter represents a gene in the chromosome, as shown by table 3. Thus, each chromosome comes in the form of a binary string of 58 bits. Binary substrings corresponding to learning rate and Momentum term genes of a chromosome are converted into decimal values, then normalized using the Min-Max normalization method to obtain values between 0 and 1, which will be serve as Learning rate and Momentum term of the IDS produced based on that chromosome.
- **Fitness Function or Evaluation Function:** We have opted the AUC metric [55] as a score (fitness function) of individuals of ISAGA to evaluate their goodness and suitability to the optimization problem. The AUC is a performing measuring of IDSs, which represents the capability to avoid misclassifications of network packets. In our opinion, it is a good compromise between DR (Detection Rate) metric and FPR (False Positive Rate) metric. Indeed, this is due to reason that AUC is the arithmetic mean of DR and TNR (1-FPR) as displayed by equation 3 of the AUC. As it is known, a good IDS is one that reaches a high detection rate (DR) and a low positive rate (FPR). As demonstrated by equation 3, as the value of the DR measure rises and that of FPR measure declines, Accordingly, the value of AUC augments. Thereby, from our point of view, AUC is the best metric for assessing an IDS. That is the reason of preference of AUC as fitness function.

Table 3. Structure of chromosome of ISAGA and some possible values of its values

Genes	Number of bits to encode the gens	Possible/number of values
Normalization	01	0 (Min-Max normalization) or 1 (Statistical normalization)
Activation function	01	0 (Hyperbolic tangent) or 1 (Sigmoid)
Number of nodes in hidden layer 01	08	256
Number of nodes in hidden layer 02	08	256
Learning rate	20	2^{20} values
Momentum rate	20	2^{20} values

$$AUC = \frac{(DR + TNR)}{2} = \frac{(DR + (1 - FPR))}{2} \quad (3)$$

The fourth phase: As outlined by figure 3, ISAGA process commences with an arbitrarily generated population of 1000 individuals (potential solutions) represented by their chromosomes; each chromosome consists in a binary string of 58 bits. Afterwards, this population evolves over multiple generations by way of genetic operations such elitism, selection, recombination (crossover) and self-adaptive mutation through adaptive mutation algorithm (AMA) till stopping or optimization criterion of ISAGA is satisfied. At each generation, for every chromosome, the Fitness Hash Table (FHT) is explored to verify if this chromosome is earlier visited, in this case, its fitness value is retrieved from FHT. Otherwise, this chromosome is employed to generate an instance of an IDS based on DNN. Subsequently, this IDS firstly passes through the learning stage, later shifts to the

testing stage and returns the values of performance metrics computed at the close of last phase. Among those performance metrics, we choose the most relevant of them, namely AUC metric to serve as "Fitness Function" for assessment of goodness of chromosomes, and the AUC (fitness value) is saved in FHT. From one generation to the next, ISAGA converges towards the global optimum with the help of genetic operations mentioned formerly. At last, the best individual (chromosome) is picked out as the ultimate result once the optimization criterion is satisfied. In our work, stopping condition opted for ISAGA is creation of 200 generations. Thus, the best chromosome procured corresponds to the optimal or near-optimal values of parameters employed to construct an ideal IDS based DNN, which turns elevated detection rate and lowly false alarm rate. In our optimization framework based on ISAGASAA module, ISAGA uses the following algorithms/methods:

- Elitism.
- Roulette Wheel Selection.
- Single point Crossover.
- Adaptive Mutation Algorithm.
- Bit Flip Mutation.

4.2 Role of simulated annealing algorithm in the proposed system

The purpose of employment of Simulated Annealing Algorithm in our framework of optimization is to optimize Improved Self-Adaptive Genetic Algorithm (ISAGA) process. Simulated Annealing Algorithm is a hill climbing algorithm that initially admits worse solutions at a tall rate, afterwards as the algorithm runs; it progressively lessens the rate in which worse solutions are accepted. One of the easiest manners to implement this characteristic into a genetic algorithm is by updating the crossover rate, to commence with a high rate then gradually decline the rate of crossover as the algorithm proceeds. This initial high crossover rate will push ISAGA to search a wide area of the search space. Subsequently, as the crossover rate is leisurely decreased, ISAGA should begin to concentrate its search on regions of the search space where fitness values are higher.

As shown by figure 3, to vary the crossover rate/probability, we have utilized a temperature variable named "**Temp-SAA**", which begins high, or "hot", and slowly diminishes, or "cools" with the aid of a Cool rate function as the algorithm advances. This heating and cooling approach is directly inspired by the process of annealing found in metallurgy. On completion of each iteration/generation of ISAGA, the temperature is cooled slightly, which declines the crossover rate that will be employed in the following generation of ISAGA [20].

4.3 Framework of our proposed MLANIDS

In this section, with the objective of clarification of functioning of our system, we utilize Kyoto dataset version 2015 as IDS dataset. Though, experiments has been carried out with two datasets, namely, Kyoto dataset version 2015 and CIDDS-001 dataset.

Our system "MLANIDS" goes initially through an optimization stage by the means of ISAGASAA framework, with the aim to find the ideal or near-ideal values of the parameters employed to develop an optimal IDS based DNN. Accordingly, it becomes mature to run in detection mode. The framework of our system in optimization mode embraces four modules as displayed in figure 1.

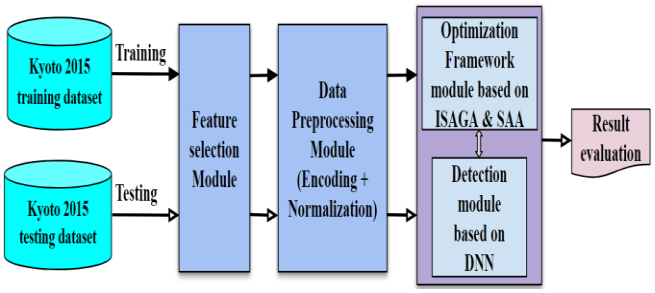

Figure 1. Framework of MLANIDS in optimization stage

- **Feature selection module:** Feature selection is the most pivotal step in developing intrusion detection models. Our intrusion detection model encompasses a feature selection module principally to pick out fruitful features for intrusion detection. This module affords selection of a set of 14 pertinent features among 24 features of Kyoto dataset version 2015 (subsection 6.2).
- **Data preprocessing module:** Data Preprocessing comprise two actions; data conversion (Categorical encoding) and Normalization. "Categorical encoding" mentions to the process of assigning numeric values to nonnumeric features/attributes, thus as to turn the processing task much easier, as numeric data can be readily treated upon. While, "Normalization or Scaling" refers to the procedure of scaling the feature values to a small range that can assist to get better detection outcomes and avoid numerical difficulties in the course of the computation. Our data-preprocessing module employs Min-Max normalization and Statistical normalization techniques.
- **Detection module based on DNN and optimization module based on ISAGA & SAA:** Detection module based on DNN interacts with an optimization module based on ISAGA & SAA as clarified in detail in subsection 4.1 with the aim to seek optimal or near-optimal values of parameters used to develop an ideal IDS based DNN. The phase of optimization is called "optimization stage". This phase is accomplished at the end of ISAGA process. Thus, the optimal values of the parameters served to build an IDS based DNN are found.

Subsequent to passing through optimization stage and uncovering the ideal parameters to create an IDS based DNN, the ideal MLANIDS works in detection mode as shown in figure 2, to forecast the class of the given records or packets retrieved from Kyoto dataset. In case of benign or legitimate sample, it is permitted to access to Cloud infrastructure, else it is discarded and the alert system module is warned. In the detection phase, we have added to our system a supplementary module, namely "Alert System".

- **Alert System:** Triggers alarms about intrusions that are identified by optimized detection module based on DNN, and transmits them to the security administrator for notification and further investigation.

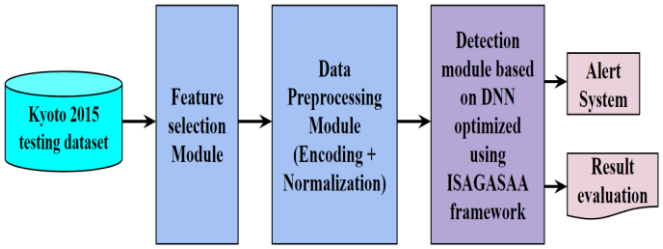

Figure 2. Framework of MLANIDS in detection mode

5. Strategic Positioning of the Proposed System in a Cloud Network

The purpose of the developed IDS is to uncover intruders and malevolent actions in and around the cloud computing environment by surveillance of network traffic, hence preserving availability, confidentiality, integrity and performance of cloud assets and afforded services. It allows detecting and blocking assaults in real time damaging the security of the Cloud Datacenter.

As shown in figure 4, we suggest implementing our NIDS on two strategic and critical locations:

- **Front-End of Cloud:** Positioning NIDS on front end of Cloud aids to detect network intrusions or assaults emanate from external network of Cloud, undertaken from zombie hosts or by attackers connected to the Internet who try to detour the firewall with the intention of accessing the internal cloud, which can be a private one. Thereby, NIDS acts as the second line of protection behind the firewall to remedy its flaws [56], and represents an additional preventive layer of security [57].
- **Back-End of Cloud:** Placing NIDS sensors on processing servers situated at back end of Cloud facilitates uncovering of intrusions happening on its internal network. In a virtual environment, we have plenty virtual machines on the same physical server, and they can inter-communicate through the medium of the virtual switch without leaving the physical server. Consequently, network security appliances on the LAN can't watch this network traffic; if the traffic does not need to fit through those devices mainly a firewall, as a result, a loophole for all types of security attacks will be opened. Thus, the departure point of an attacker/hacker is compromising only one VM, and using it as a springboard to gain control of the other VMs within the same hypervisor. This is typically done without being watched or detected, providing the attacker an enormous hack domain. Furthermore, the virtual environment is exhibited to diverse threats and risks, focused primarily on the hypervisor: VM escape, Hyper jacking, VM migration, Inter-VM traffic, and VM theft.

Our NIDS is conceived to watch that virtual traffic, and as well the flow of traffic from or to the processing server on the physical network. We haven't opted to implement the NIDS on every virtual machine since it will be a supplementary burden; it will weigh down the work of the VM. Moreover, such configuration necessitates several instances of NIDS, which makes complex management of NIDS while VMs are dynamically shifted, provisioned or de-provisioned.

6. Experimentation and Discussion

The experiences have been performed using a Windows 10 – 64 bits PC with 32 GB RAM and Intel(R) Core-i7 2700K CPU. For simulation, we have utilized CloudSim simulator 4.0. In the first subsection of the present section, we provide the performing measurings used for appraisal of our system, followed by introduction in the second subsection of data preprocessing techniques employed in our study. Thereafter, the third subsection presents the optimization framework ISAGASAA used in our work. Next, the fourth subsection describes in detail the primary dataset used in our work that is to say Kyoto dataset version 2015 for implementation and validation of our system, followed by experimental outcomes attained. Besides, assessment of our model on other IDS dataset was fulfilled, thus, we provide the results of the experiments conducted on CIDD-001 dataset in the fifth subsection. At last, in the six subsection, we give a benchmarking study between efficiency or performance of our model and other modern state-of art methods in the light of experimental outcomes reached.

6.1 Performance measurements

The performance of an intrusion detection system is evaluated by its ability to give a correct classification of events to be an attack or a normal behavior. According to the real nature of a given event and the prediction of an IDS, we found four possible outcomes that can be understood by the confusion matrix as given in the table 4 [58].

Table 4. Confusion matrix

Actual class	Predicted class	
	Attack	**Normal**
Attack	True positive (TP)	False negative (FN)
Normal	False positive (FP)	True negative (TN)

- **True Positive (TP):** Number of instances correctly predicted as attacks.
- **False Positive (FP):** Number of instances wrongly predicted as attacks.
- **True Negative (TN):** Number of instances correctly predicted as non-attacks (normal instances).
- **False Negative (FN):** Number of instances wrongly predicted as non-attacks [59].

The above four performance metrics in addition to other also performance measurements illustrated by equations 4 to 11, which are calculated using the confusion matrix, are used to evaluate the IDS effectiveness. Generally, a good Intrusion detection system (IDS) yields high accuracy and detection rate as well as low false positive rate.

$$Accuracy\,(ACC) = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$Detection\ \ Rate(DR / \mathrm{Re}call) = \frac{TP}{TP + FN} \quad (6)$$

$$False\ \ Alarm\ \ Rate\ \ (FAR) = \frac{FP}{FP + TN} \quad (7)$$

$$F\text{-}\ score = \frac{2' \ \mathrm{Re}call' \ \mathrm{Precision}}{\mathrm{Re}call + \mathrm{Precision}} \quad (8)$$

$$AUC = 0.5' \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \quad (9)$$

$$True\ \ Negative\ \ Rate\ \ (TNR) = \frac{TN}{TN + FP} \quad (10)$$

$$False\ Negative\ Rate\ (FNR) = \frac{FN}{FN + TN} \quad (11)$$

6.2 Data preprocessing

In our study, training and testing subsets used in our experiments are retrieved from the two IDS datasets employed, namely Kyoto dataset and CIDDS-001 dataset. With the purpose to make the records included in the subsets mentioned beforehand, ready to be processed by our proposed model, they must first be handled and prepared with the help of the subsequent two operations:

- **Numericalization or categorical encoding:** This action refers to the process of assigning numeric values to nonnumeric features/attributes, thus as to turn the processing task much easier, as numeric data can be readily treated upon [18].
- **Normalization:** The attributes with elevated values can dominate the outcomes than the attributes with lower values .This dominance can be mitigated by the technique of normalization, i.e., scaling the values within certain range. Normalization is defined as is the procedure of enclosing the values of attributes to a peculiar range to minimize the complexity involved in processing data spread over an absolute range and type of values. Diverse attributes have values spread over large ranges and types. For that reason, they are to be reduced to a particular range being helpful to permit correct processing and analysis of the data [43]. To normalize data, the mean-range [0, 1] (Min-Max) normalization and Statistical normalization (Z-score) methods are employed. The basis of choice of these approaches is that they offer better outcomes in terms of time and classification rate [18].

    1. **Mean range [0, 1] (Min-Max normalization):** As indicated by equation 12, the mean range method normalizes an attribute value by subtracting minimum value of that attribute from the present value. This value is then divided by the gap between maximum and minimum values of that attribute. X and X' are value to be normalized and the normalized attribute value respectively. MinA and MaxA are the minimum and maximum possible values for attribute A before normalization.

    $$X' = \frac{X - MinA}{MaxA - MinA} \quad (12)$$

    2. **Statistical normalization (Z-score normalization):** The value X of an attribute A is transformed in X' according to formula (13). μ is the mean and α is standard deviation of given attribute.

    $$X' = \frac{X - m}{a} \quad (13)$$

6.3 Optimization framework ISAGASAA

Table 5 exhibits the parameters of our hybrid optimization framework used in this work with the aim to construct intelligently and automatically an ideal or near ideal IDS based on DNN. This framework combines our Improved Self-Adaptive Genetic Algorithm and Simulated Annealing Algorithm, and it is used posteriorly throughout the experiments conducted with the Kyoto version 2015 and CIDDS-001 datasets.

Table 5. Parameters of our hybrid optimization framework based ISAGASAA

Components of the framework ISAGASAA	Parameters	Value
Improved Self-Adaptive Genetic Algorithm	Length of chromosomes	58 bits
	Elitism number: the number of best chromosomes which will be copied without changes to a new population (next generation)	100
	Population size	1000
	Maximum number of generations	200
	Initial Crossover rate (Dynamic parameter)	0.95
	Initial Mutation rate (Dynamic parameter)	0.1
	Size of Fitness Hash Table (FHT)	10000
Simulated Annealing Algorithm (SAA)	Initial Temperature (Dynamic parameter)	1.0
	Cooling rate	0.001

6.4 Experimentation based on kyoto university benchmark dataset

6.4.1 Description of Kyoto Dataset Version 2015

Kyoto 2006+ University Benchmark Dataset [60] is one of the most freshly used intrusion detection dataset in the IDS arena [46]. It consists of real network traffic data originally gathered between November 2006 to December 2015 in Kyoto University. The Kyoto dataset is captured using honeypots, darknet sensors, email server and web crawler. Song et al. in [45] provided a detailed analysis of honeypots (i.e. computer network security mechanisms that detect attempts of unauthorized use of information) and darknets data collected on many real and virtual machines as honeypots. They have deployed various types of honeypots, darknets and other systems on the five networks inside and outside of the Kyoto University, and collected all traffic data to and from 348 honeypots (Windows XP, Windows Server, Solaris…). All traffic was thoroughly inspected using three security software SNS7160 IDS, Clam AntiVirus, Ashula and since Apr. 2010, snort was added. This dataset was established to surmount the issues of underperformance of machine learning based IDS model trained and tested on KDD or NSL-KDD and any other related old datasets considerably criticized for not reflecting the present trend of network situation and sophistication of ever evolving cyber-attacks [53].

Kyoto dataset is a multivariate attributes dataset that comports 24 statistical features pulled out from captured data, comprising 14 conventional features that were derived from KDD Cup 99 benchmark dataset, and 10 supplementary features added for analysis and further investigation. This work employs subset of December 31, 2015 and has used the first 14 features (conventional features) that are convenient for network-based IDSs [54], and the label that points out whether the record is an attack or normal, and excluded the features that can be

used to investigate what types of attacks arisen on computer networks. Table 14 shows selected features used in our experiments. Label feature indicates whether the session was attack or not; '1' means normal. '-1' means attack was observed in the session. The data pertaining to date **December 31, 2015 of Kyoto dataset** is used for this study and this dataset has 309068 records, out of which 23062 are normal and the 286006 are attack data records.

6.4.2 Experimental Results

As highlighted beforehand in subsection 6.4.1, the data pertaining to date **December 31, 2015 of Kyoto dataset** was used in this work. This dataset contains two classes of network connections; 23062 normal records and 286006 attack records. Each class is splited randomly to two sets using a configuration of 60% for training and 40% for testing in order to build training dataset and testing dataset used in our experiences as shown by table 6.

Table 6. Distribution and size of training and testing datasets Kyoto dataset version 2015

Datasets	Description	Intrusive Records	Normal Records	Total Records
Dataset 20151231	Full Kyoto dataset of 2015-12-31	286006	23062	309068
Training dataset	60% of Kyoto dataset of 2015-12-31	171604	13838	185442
Testing dataset	40% of Kyoto dataset of 2015-12-31	114402	9224	123626

The experiments carried out on our model reveal that at the end of running of our framework ISAGASAA, that is to say after 200 generations, the best individual (chromosome) found leads to construction of the best MLANIDS called "MLANIDS_ KYOTO2015". Tables 7 and 8 illustrate respectively configuration and performances attained by that ANIDS.

Table 7. Configuration of best MLANIDS_ KYOTO2015

Configuration Parameter	Value
Number of nodes in Input layer	14
Number of nodes in Hidden layer 01	10
Number of nodes in Hidden layer 02	7
Number of nodes in Output layer	1
Activation function	Sigmoid
Normalization of data	Min-Max
Learning rate	8.347698102564871E-7
Momentum term	1.143019854621969E-4

Table 8. Performance of best MLANIDS_ KYOTO2015

Performance Metric	Value
Accuracy	99.95%
Precision	99.99%
Detection Rate (DR)	99.95%
False Negative Rate (FNR)	0.05%
False Positive Rate (FPR)	0.02%
True Negative Rate (TNR)	99.98%
F-score	0.99
AUC (Ability to avoid misclassifications)	99.96%

6.5 Experimentation based on CIDDS-001 dataset

6.5.1 Description of CIDDS-001 Dataset

CIDDS-001 (Coburg Network Intrusion Detection Dataset) [61] is a labelled flow based dataset created by M. Ring et al. [62] in a Cloud environment based on OpenStack platform. This dataset holds unidirectional NetFlow data. It consists of traffic data from two server's i.e. OpenStack and External server.

Table 9. Attributes within the CIDDS-001 data set

Nr.	Feature	Description
1	Src IP	Source IP Address
2	Src Port	Source Port
3	Dest IP	Destination IP Address
4	Dest Port	Destination Port
5	Proto	Transport Protocol (e.g. ICMP, TCP, or UDP)
6	Date first seen	Start time flow first seen
7	Duration	Duration of the flow
8	Bytes	Number of transmitted bytes
9	Packets	Number of transmitted packets
10	Flags	OR concatenation of all TCP Flags
11	Class	Class label (normal, attacker, victim, suspicious or unknown)
12	AttackType	Type of Attack (PortScan, DoS, Bruteforce, PingScan)
13	AttackID	Unique attack id. All flows which belong to the same attack carry the same attack id.
14	AttackDescription	Provides additional information about the set attack parameters (e.g. the number of attempted password guesses for SSH-Brute-Force attacks)

CIDDS-001 dataset is produced by emulating small business environment which consist of OpenStack environment having internal servers (web, file, backup and mail) and an External Server (file synchronization and web server) which is

deployed on the internet to capture real and up-to-date traffic from the internet. It includes three logs files (attack logs, client configurations and client logs) and traffic data from two servers where each server traffic comprises of 4 four week captured traffic data [35]. CIDDS-001 embraces realistic normal and attack traffic allowing substantial benchmarking of network intrusion detection systems in a Cloud environment. It contains 14 attribute, the first 10 attributes are the default NetFlow attributes and the last four attributes are additional attributes, names and descriptions of features are tabulated in table 9. A total of 32 million of normal and attack flows are captured in the dataset within four weeks. Ring et al. [62] have exploited 92 types of attacks to create this dataset. This makes it a significant benchmark for intrusion detection systems. The reasons above motivated us to use this dataset in this article. As outlined antecedently, the CIDDS-001 has 14 attributes out of which the first 11 features have been used in this study.

6.5.2 Experimental Results

The original version of CIDDS-001 comprises five classes, i.e. normal, suspicious, unknown, attacker, and victim, and since our goal is to appraise anomaly based IDS, we only incorporated normal and attacker classes in our experimental dataset. Thereby, as displayed by table 10, the reduced version of CIDDS-001 dataset employed in this work holds 953298 normal instances and 65652 attacker instances, retrieved from both week 1 and week 2 of OpenStack and ExternalServer traffic folders respectively.

Table 10. Construction of a reduced version of CIDDS-001 dataset

Traffic folder	File source	Normal Records	Attack Records
OpenStack	CIDDS-001-internal-week1.csv	924862	63263
ExternalServer	CIDDS-001-external-week2.csv	28436	2389
Reduced CIDDS-001 dataset		953298	65652

This reduced version of CIDDS-001 dataset is subsequently splited into train and test subsets using a configuration of 60% for training and 40% for testing. Table 11 exhibits distribution and size of those subsets.

Table 11. dataset Distribution and size of testing and training CIDDS-001 subsets

Datasets	Normal Records	Intrusive Records	Total Records
Training subset	381319	924862	63263
Testing subset	571979	28436	2389

The experiments conducted on our model point that on completion of executing of our framework ISAGASAA, that is to say after 200 generations, the best individual (chromosome) found allows building the best MLANIDS, called “MLANIDS_ CIDDS-001”. Tables 12 and 13 display respectively configuration and performances achieved by that IDS.

Table 12. Configuration of best MLANIDS_ CIDDS-001

Configuration Parameter	Value
Number of nodes in Input layer	10
Number of nodes in Hidden layer 01	7
Number of nodes in Hidden layer 02	5
Number of nodes in Output layer	1
Activation function	Sigmoid
Normalization of data	Min-Max
Learning rate	8.358769540214567E-7
Momentum term	1.158746562046783E-4

Table 13. Performance of best MLANIDS_ CIDDS-001

Performance Metric	Value
Accuracy	99.96%
Precision	99.47%
Detection Rate (DR)	99.95%
False Negative Rate (FNR)	0.05%
False Positive Rate (FPR)	0.04%
True Negative Rate (TNR)	99.96%
F-score	0.99
AUC (Ability to avoid misclassifications)	99.95%

6.6 Comparison with related works and discussion

The subsections 6.4 and 6.5 give the results of implementation and assessment of our model, which consists of building automatically an ANIDS based DNN through our hybrid optimization framework ISAGASAA (combination of improved Self-Adaptive Genetic Algorithm (ISAGA) and Simulated Annealing Algorithm (SAA)), with the help of two different IDS datasets namely Kyoto version 2015 and CIDDS-001 2017 respectively. As result, we have obtained the two best Machine Learning ANIDSs (MLANID) according to the datasets utilized, which are MLANIDS_KYOTO2015 and MLANIDS_CIDDS-001 described by tables 8 and 13 respectively. Afterwards, we have compared those ANIDS between them, and with varied state-of-art approaches.

Analysis of the tables 8, 13 and 15 together with figures 5-8 leads to the following conclusions:

- Our ANIDSs obtained; MLANIDS_KYOTO2015 and MLANIDS_CIDDS-001 attain approximately similar performances, and all of them yield high detection rate, high accuracy, great precision, elevated AUC and low false positive rate.
- Both MLANIDS_KYOTO2015 and MLANIDS_CIDDS-001 attain higher detection rate (99.95%), higher AUC (99.96% and 99.95%), higher accuracy (99.95% and

99.96%) and lower false positive rate (0.02% and 0.04%). Whereas the second rank is occupied by the work of Zhang et al. [63], thanks to values of DR, accuracy and F-score that are 99.92%, 99.91% and 0.99 respectively.

- The performance comparison demonstrates that our model outperforms all the 11 approaches based DNN involved in the comparison, and proves that it is the most suitable for detecting miscellaneous attacks with elevated detection rate and lowly false alarm rate.
  - Using of an improved Self-Adaptive Genetic Algorithm (ISAGA), further optimized by Simulated Annealing Algorithm (SAA) enables to explore smartly and efficaciously the search space in order to found the best set of values of parameters, which are involved in creation of ANIDS based on Deep Neural Network. Hence, avoiding us to regulating or tuning those parameters manually by way of the technique "trial and error".

Further, table 16 that is earmarked to comparison of performing of our model to 20 state-of- the-art research based on other approaches different to ANN, proves clearly that our model also surpasses those 20 works in terms of intrusion detection effectiveness. We highlight that the IDSs developed by Ibrahim & Zainal [29] and Hatef et al. [64] achieve well performances closer to our model. In fact, they reached high values of (Detection rate, Accuracy), namely (99.70% , 99.60%) and (99.38% , 99.38%) respectively.

We noticed also that implementation and incorporation of optimization strategies to the fitness function of ISAGA, namely Parallel Processing and Fitness Value Hashing have brought multiple benefits. These advantages are: an average of 90% diminution of execution time compared to a standard GA, acceleration of the ISAGA convergence process and save of processing power.

With the aim to render our proposed IDS model reactive, autonomous and able to protect in real-time the cloud resources against attacks, we have integrated to it a reactive functionality. It is the capacity to discard the malicious records or packets automatically and in real-time. As explained in subsection 4.3, during detection mode, after classifying a packet by the detection module and in case of intrusion, the malicious packet is deleted and the "Alert System" module is notified. This later module afterwards send an alert to the security administrator for warning and further investigation.

Our work offers the following novel contributions:

- Our proposed model enables comparing implicitly shallow and deep neural networks. Given that the model of chromosomes used by our ISAGA contains the gene termed “Nb of nodes in hidden layer 02” encoded in 8 bits, therefore, a possible solution (chromosome) may be shallow neural network (SNN) or deep neural network (DNN). If the value of this gene is equal to 0 the resulted neural network (NN) is an instance of SNN, otherwise we get DNN. Certainly, during running of ISAGASAA, it had explored the search space and generated several instances of both SNN and DNN through its generic operations mainly crossover and mutation. Consequently, at the end of its execution, it found the fittest chromosome according to the dataset used, allowing building the best ANIDSs based on DNN as shown by the tables 8 and 13. Owing of the obtained outcomes, we conclude that DNN can yield better performance than SNN. As confirmed by the works [24, 65, 66, 67], a good tuning of DNN parameters lead to reach high performances. In our proposed system, that tuning is automatically performed by ISAGASAA thanks to two characteristics, namely exploration of the search space and exploitation of genetic patrimony transferred from generation to the next. The two mentioned features are insured by genetic operations of ISAGA. Moreover, our conclusion is supported by the two books entitled "Deep learning" [68] and "Neural Network Design" [69], which state that the trend today is to design artificial NN with more hidden layers, therefore deep neural networks.
- In our literature review carried out with the aim to determinate uses of Genetic Algorithm (GA) with Artificial Neural networks (ANN), either shallow or deep neural networks in the field of Network IDSs (NIDS), we have found mainly two main categories of uses. The first one is use of GA for generation of rules of intrusion detection (Rule Generation) for IDS based ANN, whereas the second use is selection of the best/relevant attributes (Feature Selection) for detection of anomalies/attacks. As far as we know, our work is the first research project that employs a variant of GA, namely ISAGA, in such manner that consists of seeking the optimal or near optimal values of parameters involved in construction of IDS based DNN or affecting its performance with the help of ISAGA, optimized by using SAA.
- To the best of our knowledge and according to our survey, to build an IDS based on DNN, in general, researchers choose randomly the values of the parameters implicated in building of DNN or follow trial and error method to determine those values. However, in our work, we use a machine-learning optimization framework that combines our variant of standard GA called ISAGA "Improved Self Adaptive Genetic Algorithm", described previously in detail and “Simulated Annealing Algorithm” (SAA), to explore automatically and intelligently the search space of those values in order to find their optimal values. Those ideal values allow to construct a perfect or near perfect ANIDS based DNN. In fact, the two MLANIDS built with the aid of the two IDS datasets used, yield higher performances in comparison to 34 IDS spread over the tables 15 and 16.

7. Conclusions and Future Work

With the purpose to preserve the confidentiality, integrity and availability (CIA) of the networks and information systems in the cloud environment, it is inevitable to develop a powerful Network IDS (NIDS), which will act as a security guard and protect cloud resources and services against inside and outside assaults. In fact, such NIDS should attain higher performance in terms of intrusion detection; it must ensure higher detection rate, higher accuracy, higher precision and lower false positive rate with an affordable computational cost. Hence, to create a NIDS that overcomes the above issues and satisfies the requirements brought up earlier, we have adopted a clever approach to build automatically the expected and hoped NIDS based on Deep Neural Network (DNN). Our methodology relies on using a novel hybrid optimization framework termed “ISAGASAA” that combines our developed self-adaptive heuristic search algorithm called “Improved Self-Adaptive Genetic Algorithm (ISAGA)”, and

Simulated Annealing Algorithm (SAA) with the view of seeking for the ideal values of the parameters involved in building of IDS based DNN (IDSDNN) or influencing its performance. Thereafter, leveraging the found values to build a perfect or near-perfect NIDS based DNN. ISAGA is our variant of standard Genetic Algorithm (GA), which is developed based on GA, improved through an Adaptive Mutation Algorithm (AMA) and optimization strategies, namely Parallel Processing and Fitness Value Hashing. Further, SAA is incorporated to ISAGA with the aim to optimize its heuristic search. In ISAGA process, we have employed binary encoding for chromosomes, while AUC metric was adopted as a fitness function (score) for appraisal of the goodness or adaptability of the chromosomes produced versus the optimization problem in hand. In every generation of ISAGA, each chromosome generated is utilized to develop an instance of IDSDNN, which subsequently goes through learning stage and a test stage. At the close of the former stage, AUC measure is calculated. ISAGA process commences with an arbitrarily generated population, which evolves by means of elitism, selection, recombination (crossover) and self-adaptive mutation. Lastly, the best individual (chromosome) is choose as the ultimate result once the optimization criterion is fulfilled. In our work, termination condition adopted for ISAGA is creation of 200 generations. Thereby, at the conclusion of ISAGA process, the ideal or near-ideal values of parameters used to construct an optimal IDSDNN are found, which enables building an efficient and effective machine learning anomaly NIDS called "MLANIDS" achieving elevated detection rate and lowly false positive rate.

Experimental outcomes obtained by using CloudSim 4.0 and two IDS datasets, that is to say Kyoto version 2015 and CIDDS-001 2017 prove that our two generated MLANIDSs, outperform numerous state-of-the-art approaches. Besides, performance enhancement strategies integrated to ISAGA have diminished execution time, convergence time and saved computational power.

We have chosen to position our proposed IDS on Front-End and Back-End of the Cloud, to uncover and stop intrusions in real time damaging the security of the Cloud Datacenter.

We project in our forthcoming work to utilize other meta-heuristic algorithms (such artificial bee colony (ABC) algorithm, particle swarm optimization, whale optimization algorithm, crow search algorithm or ant colony optimization (ACO) to compare them with ISAGA Algorithm developed in this study.

8. Acknowledgement

We would like to thank all members of Computer Science and Systems Laboratory within Department of Mathematics and Computer in Faculty of Sciences Ain Chock, Hassan II University for precious help and permanent support.

References

[1] S. Ravji, M. Ali, "Integrated Intrusion Detection and Prevention System with Honeypot in Cloud Computing," IEEE 2018 International Conference on Computing, Electronics & Communications Engineering (ICCECE), Southend, UK, pp. 95-100, 2018.

[2] The NIST definition of cloud computing, 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[3] G. Brunette, R. Mogull, "Security guidance for critical areas of focus in cloud computing v2. 1," Cloud Security Alliance, pp. 1-76, 2009.

[4] M. M. Sakr, M. A. Tawfeeq, A. B. El-Sisi, "Network Intrusion Detection System based PSO-SVM for Cloud Computing," International Journal of Computer Network and Information Security, Vol. 10, No. 3, pp. 22-29, 2019.

[5] T. Nathiya, G. Suseendran. "An Effective Hybrid Intrusion Detection System for Use in Security Monitoring in the Virtual Network Layer of Cloud Computing Technology," In: V. Balas, N. Sharma, A. Chakrabarti (Eds.), Data Management, Analytics and Innovation. Advances in Intelligent Systems and Computing, Vol. 839, Springer, Singapore, Singapore, pp. 483-497, 2019.

[6] ORACLE and KPMG Enterprises, "2020 ORACLE and KPMG Cloud Threat Report," 2020. https://www.oracle.com/a/ocom/docs/cloud/oracle-cloud-threat-report-2020.pdf.

[7] L. Khatibzadeh, Z. Bornaee, A. Ghaemi Bafghi, "Applying Catastrophe Theory for Network Anomaly Detection in Cloud Computing Traffic," Security and Communication Networks, Vol. 2019, pp. 1-11, 2019.

[8] Symantec Enterprise, "2019 Internet Security Threat Report," Mountain View, CA, USA, 2019. https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf.

[9] Netskope Enterprise, "2020 Netskope Cloud and Threat Report," Santa Clara, CA, USA, 2021. https://resources.netskope.com/cloud-reports/cloud-and-threat-report-july-2021.

[10] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, M. Rajarajan, "A survey of intrusion detection techniques in cloud," Journal of Network and Computer Applications, Vol. 36, No. 1, pp. 42-57, 2013.

[11] Y. Gao, Y. Liu, Y. Jin, J. Chen, H. Wu, "A Novel Semi-Supervised Learning Approach for Network Intrusion Detection on Cloud-Based Robotic System," IEEE Access, Vol. 6, pp. 50927-50938, 2018.

[12] P. Ghosh, A. Karmakar, J. Sharma, S. Phadikar, "CS-PSO based Intrusion Detection System in Cloud Environment," In: A. Abraham, P. Dutta, J. Mandal, A. Bhattacharya, S. Dutta (Eds.), Emerging Technologies in Data Mining and Information Security, Advances in Intelligent Systems and Computing, Vol. 755, Springer, Singapore, Singapore, pp. 261-269, 2019.

[13] M. Idhammad, K. Afdel, M. Belouch, "Distributed Intrusion Detection System for Cloud Environments based on Data Mining techniques," Procedia Computer Science, Vol. 127, pp. 35-41, 2018.

[14] S. M. Mehibs, S. H. Hashim, "Proposed Network Intrusion Detection System In Cloud Environment Based on Back Propagation Neural Network," Journal of University of Babylon for Pure and Applied Sciences, Vol. 26, No. 1, pp. 29-40, 2018.

[15] W. Yassin, N. I. Udzir, Z. Muda, A. Abdullah, M. T. Abdullah, "A cloud-based intrusion detection service framework," 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), Kuala Lumpur, Malaysia, pp. 213-218, 2012.

[16] S. X. Wu, W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," Applied soft computing, Vol. 10, No. 1, pp. 1-35, 2010.

[17] M. AL-Shabi, "Design of a Network Intrusion Detection System using Complex Deep Neuronal Networks," International Journal of Communciation Networks and Information Security (IJCNIS), Vol. 13, No. 3, pp. 409-415, 2021.

[18] Z. Chiba, N. Abghour, K. Moussaid, M. Rida, "A novel architecture combined with optimal parameters for back propagation neural networks applied to anomaly network intrusion detection," Computers & Security, Vol. 75, pp. 36-58, 2018.

[19] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A. R. Mohamed, N. Jaitly,, ... & B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," IEEE Signal processing magazine, Vol. 29, No. 6, pp. 82-97, 2012.

[20] L. Jacobson, B. Kanbe, "Genetic algorithms in Java basics," Apress, New York, USA, 2015.

[21] D. E. Kim, M. Gofman, "Comparison of shallow and deep neural networks for network intrusion detection," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, pp. 204-208, 2018.

[22] J. H. Woo, J. Y. Song, Y. J. Choi, "Performance Enhancement of Deep Neural Network Using Feature Selection and Preprocessing for Intrusion Detection," 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Okinawa, Japan, pp. 415-417, 2019.

[23] K. G. Sheela, S. N. Deepa, "Review on methods to fix number of hidden neurons in neural networks," Mathematical Problems in Engineering, Vol. 2013, pp. 1-11, 2013.

[24] Z. Zhang, G. Zhang, Y. Shen, Y. Zhu, "Intrusion Detection Model Based on GA Dimension Reduction and MEA-Elman Neural Network," In: L. Barolli, F. Xhafa, N. Javaid, T. Enokido (Eds.), Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2018), Advances in Intelligent Systems and Computing, Vol. 773, Springer, Cham, Switzerland, pp. 354-365, 2019.

[25] K. K. Ghanshala, P. Mishra, R. C. Joshi, S. Sharma, "BNID: A Behavior-based Network Intrusion Detection at Network-Layer in Cloud Environment," 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India, pp. 100-105, 2018.

[26] S. I. Shyla, S. S. Sujatha, "Cloud Security: LKM and Optimal Fuzzy System for Intrusion Detection in Cloud Environment," Journal of Intelligent Systems, Vol. 29, No. 1, pp. 1626-1642, 2019.

[27] N. M. Ibrahim, A. Zainal, "A Distributed Intrusion Detection Scheme for Cloud Computing," International Journal of Distributed Systems and Technologies (IJDST), Vol. 11, No. 1, pp. 68-82, 2020.

[28] M. Rabbani, Y. L. Wang, R. Khoshkangini, H. Jelodar, R. Zhao, P. Hu, "A hybrid machine learning approach for malicious behaviour detection and recognition in cloud computing," Journal of Network and Computer Applications, Vol. 151, pp. 1-13, 2020.

[29] N. Neha, M.G. Raman, N. Somu, R. Senthilnathan, V.S. Sriram, "An Improved Feedforward Neural Network Using Salp Swarm Optimization Technique for the Design of Intrusion Detection System for Computer Network," In: A. Das, J. Nayak, B. Naik, S. Pati, D. Pelusi (Eds.), Computational Intelligence in Pattern Recognition, Advances in Intelligent Systems and Computing, Vol. 999, Springer, Singapore, Singapore, pp. 867-875, 2020.

[30] S. Krishnaveni, P. Vigneshwar, S. Kishore, B. Jothi, S. Sivamohan, "Anomaly-Based Intrusion Detection System Using Support Vector Machine," In: S. Dash, C. Lakshmi, S. Das, B. Panigrahi (Eds.), Artificial Intelligence and Evolutionary Computations in Engineering Systems, Advances in Intelligent Systems and Computing, Vol. 1056, Springer, Singapore, Singapore, pp. 723-731, 2020.

[31] M.S. Abirami, U. Yash, S. Singh, "Building an Ensemble Learning Based Algorithm for Improving Intrusion Detection System," In: S. Dash, C. Lakshmi, S. Das, B. Panigrahi (Eds.), Artificial Intelligence and Evolutionary Computations in Engineering Systems, Advances in Intelligent Systems and Computing, Vol. 1056, Springer, Singapore, Singapore, pp. 635-649, 2020.

[32] T. Thilagam, R. Aruna, "Intrusion detection for network based cloud computing by custom RC-NN and optimization," ICT Express, Vol. 7, No. 4, pp. 512-520, 2021.

[33] S. Sobin Soniya, S. Maria Celestin Vigila, "Feedback deer hunting optimization algorithm for intrusion detection in cloud based deep residual network," International Journal of Modeling, Simulation, and Scientific Computing, Vol. 12, No. 6, pp. 2150047-2150057, 2021.

[34] S. Sayed, M. Nassef, A. Badr, I. Farag, "A nested genetic algorithm for feature selection in high-dimensional cancer microarray datasets," Expert Systems with Applications, Vol. 121, pp. 233-243, 2019.

[35] R. Pereira, L. Aelenei, "Optimization assessment of the energy performance of a BIPV/T-PCM system using Genetic Algorithms," Renewable Energy, Vol. 137, pp. 157-166, 2019.

[36] J. Thompson, K.A. Dowsland, "General cooling schedules for a simulated annealing based timetabling system," In: E. Burke, P. Ross (Eds.), Practice and Theory of Automated Timetabling (PATAT 1995), Lecture Notes in Computer Science, Vol. 1153, Springer, Berlin, Heidelberg, Germany, pp. 345-363, 1996.

[37] L. R. Rere, M. I. Fanany, A. M. Arymurthy, "Simulated annealing algorithm for deep learning," Procedia Computer Science, Vol. 72, pp. 137-144, 2015.

[38] K. L. Du, M. N. S. Swamy, "Search and optimization by metaheuristics," Birkhauser, Cham, Switzerland, 2016.

[39] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, "Equation of state calculations by fast computing machines," The journal of chemical physics, Vol. 21, No. 6, pp. 1087-1092, 1953.

[40] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by simulated annealing," Science, Vol. 220, No. 4598, pp. 671-680, 1983.

[41] B. Suman, P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization," Journal of the operational research society, Vol. 57, No. 10, pp. 1143-1160, 2006.

[42] Y. Nourani, B. Andresen, "A comparison of simulated annealing cooling strategies," Journal of Physics A: Mathematical and General, Vol. 31, No. 41, pp. 8373-8385, 1998.

[43] N. Lokeswari, B. C. Rao, "Artificial Neural Network Classifier for Intrusion Detection System in Computer Network," Second International Conference on Computer and Communication Technologies. Hyderabad, India, pp. 581-591, 2016.

[44] B.A. Tama, K. H. Rhee, "Attack Classification Analysis of IoT Network via Deep Learning Approach," Research Briefs on Information & Communication Technology Evolution (ReBICTE), Vol. 3, pp. 1-9, 2017.

[45] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, K. Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, Salzburg, Austria, pp. 29-36, 2011.

[46] D.A. Musbau, J.K. Alhassan, "Ensemble Learning Approach for the Enhancement of Performance of Intrusion Detection System," International Conference on Information and Communication Technology and its Applications (ICTA 2018), Minna, Nigeria, pp. 1-8, 2018.

[47] D. Singh, D. Patel, B. Borisaniya, C. Modi, "Collaborative ids framework for cloud," Journal of Network Security, Vol. 18, No. 4, pp. 699-709, 2016.

[48] W. Wang, X. Zhang, S. Gombault, S. J. Knapskog, "Attribute normalization in network intrusion detection," IEEE 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN), Kaohsiung, Taiwan, pp. 448-453, 2009.

[49] S. Kumar, A. Yadav, "Increasing performance Of intrusion detection system using neural network," 2014 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), Ramanathapuram, India, pp. 546-550, 2014.

[50] N. Sen, R. Sen, M. Chattopadhyay, "An effective back propagation neural network architecture for the development of an efficient anomaly based intrusion detection system," 2014 International Conference on Computational Intelligence and Communication Networks, Bhopal, India, pp. 1052-1056, 2014.

[51] R. Gaidhane, C. Vaidya, M. Raghuwanshi, "Intrusion Detection and Attack Classification using Back-propagation Neural Network," International Journal of Engineering Research and Technology (IJERT), Vol. 3, No. 3, pp. 1112-1115, 2014.

[52] Description of Kyoto university benchmark data, 2006. http://www. takakura.com/Kyoto_data/BenchmarkData-Description-v5. Pdf.

[53] M. A. Jabbar, R. Aluvalu, "RFAODE: A Novel Ensemble Intrusion Detection System," Procedia computer science, Vol. 115, pp. 226-234, 2017.

[54] D. D. Protić, "Review of KDD Cup'99, NSL-KDD and Kyoto 2006+ datasets," Vojnotehnički glasnik, Vol. 66, No. 3, pp. 580-596, 2018.

[55] M. Sokolova, G. Lapalme, "A systematic analysis of performance measures for classification tasks," Information Processing & Management, Vol. 45, No. 4, pp. 427-437, 2009.

[56] Z. Chiba, N. Abghour, K. Moussaid, M. Rida, "A cooperative and hybrid network intrusion detection framework in cloud computing based on snort and optimized back propagation neural network," Procedia Computer Science, Vol. 83, pp. 1200-1206, 2016.

[57] C. N. Modi, D. R. Patel, A. Patel, R. Muttukrishnan, "Bayesian Classifier and Snort based network intrusion detection system in cloud computing," 2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12), Coimbatore, India, pp. 1-7, 2012.

[58] O. Abouabdalla, H. El-Taj, A. Manasrah, S. Ramadass, "False positive reduction in intrusion detection system: A survey," 2009 2nd IEEE International Conference on Broadband Network & Multimedia Technology, Beijing, China, pp. 463-466, 2009.

[59] U. Ali, K.K. Dewangan, D.K. Dewangan, "Distributed Denial of Service Attack Detection Using Ant Bee Colony and Artificial Neural Network in Cloud Computing," In: B. Panigrahi, M. Hoda, V. Sharma, S. Goel (Eds.), Nature Inspired Computing, Advances in Intelligent Systems and Computing, Vol 652, Springer, Singapore, Singapore, pp. 165-175, 2018.

[60] Kyoto 2006+ dataset. http://www.takakura.com/Kyoto_data.

[61] CIDDS-001 dataset. https://www.hs-coburg.de/fileadmin/hscoburg/WISENT-CIDDS-001.zip.

[62] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, A. Hotho, "Flow-based benchmark data sets for intrusion detection," 16th European Conference on Cyber Warfare and Security, Dublin, Ireland, pp. 361-369, 2017.

[63] Y. Zhang, X. Chen, L. Jin, X. Wang, D. Guo, "Network intrusion detection: Based on deep hierarchical network and original flow data," IEEE Access, Vol. 7, pp. 37004-37016, 2019.

[64] M. A. Hatef, V. Shaker, M. R. Jabbarpour, J. Jung, H. Zarrabi, "HIDCC: A hybrid intrusion detection approach in cloud computing," Concurrency and Computation: Practice and Experience, Vol. 30, No. 3, 2018.

[65] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," IEEE Access, Vol. 7, pp. 41525-41550, 2019.

[66] A. Javaid, Q. Niyaz, W. Sun, M. Alam, "A deep learning approach for network intrusion detection system," Eai Endorsed Transactions on Security and Safety, Vol. 3, No. 9, p. e2, 2016.

[67] S. Gurung, M. K. Ghose, A. Subedi, "Deep learning approach on network intrusion detection system using NSL-KDD dataset," International Journal of Computer Network and Information Security, Vol. 11, No. 3, pp. 8-14, 2019.

[68] I. Goodfellow, Y. Bengio, A. Courville, "Deep learning," MIT press, Cambridge, Massachusetts, 2016.

[69] H. B. Demuth, M. H. Beale, O. De Jess, M. T. Hagan, "Neural network design," Martin Hagan, Oklahoma State University, Stillwater, Ok, USA, 2014.

[70] C. Yin, Y. Zhu, J. Fei, X. He, "A deep learning approach for intrusion detection using recurrent neural networks," IEEE Access, Vol. 5, pp. 21954-21961, 2017.

[71] N. Shone, T. N. Ngoc, V. D. Phai, Q. Shi, "A deep learning approach to network intrusion detection," IEEE transactions on emerging topics in computational intelligence, Vol. 2, No. 1, pp. 41-50, 2018.

[72] M. Al-Qatf, Y. Lasheng, M. Al-Habib, K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," IEEE Access, Vol. 6, pp. 52843-52856, 2018.

[73] T. Ma, Y. Yu, F. Wang, Q. Zhang, X. Chen, "A Hybrid Methodologies for Intrusion Detection Based Deep Neural Network with Support Vector Machine and Clustering Technique," In: N. Yen, J. Hung (Eds.), Frontier Computing (FC 2016), Lecture Notes in Electrical Engineering, Vol. 422, Springer, Singapore, Singapore, pp. 123-134, 2018.

[74] Y. Yang, K. Zheng, C. Wu, Y. Yang, "Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network," Sensors, Vol. 19, No. 11, pp. 1-20, 2019.

[75] Y. Mehmood, M. A. Shibli, A. Kanwal, R. Masood, "Distributed intrusion detection system using mobile agents in cloud computing environment," 2015 Conference on Information Assurance and Cyber Security (CIACS), Rawalpindi, Pakistan, pp. 1-8, 2015.

[76] R. Singh, H. Kumar, R. K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," Expert Systems with Applications, Vol. 42, No. 22, pp. 8609-8624, 2015.

[77] P. Ghosh, S. Jha, R. Dutta, S. Phadikar, "Intrusion Detection System Based on BCS-GA in Cloud Environment," in: N. Shetty, L. Patnaik, N. Prasad, N. Nalini (Eds.), Emerging Research in Computing, Information, Communication and Applications (ERCICA 2016), Springer, Singapore, Singapore, pp. 393-403, 2018.

[78] M. E. Aminanto, K. I. M. HakJu, K. I. M. Kyung-Min, K. I. M. Kwangjo, "Another Fuzzy Anomaly Detection System Based on Ant Clustering Algorithm," IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E100-A, No. 1, pp. 176-183, 2017.

[79] H. H. Pajouh, G. Dastghaibyfard, S. Hashemi, "Two-tier network anomaly detection model: a machine learning approach." Journal of Intelligent Information Systems, Vol. 48, No.1, pp. 1-14, 2015.

[80] S. M. Mehibs, S. H. Hashim, "Proposed Network Intrusion Detection System Based on Fuzzy c Mean Algorithm in Cloud Computing Environment," Journal of University of Babylon, Vol. 26, No. 2, pp. 27-35, 2018.

[81] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão, M. L. Proença Jr, "Network anomaly detection system using genetic algorithm and fuzzy logic, " Expert Systems with Applications, Vol. 92, pp. 390-402, 2018.

[82] R. Sharma, S. Chaurasia, "An Enhanced Approach to Fuzzy C-means Clustering for Anomaly Detection," In: A. Somani, S. Srivastava, A. Mundra, S. Rawat, (Eds.), Proceedings of First International Conference on Smart System, Innovations and Computing, Smart Innovation, Systems and Technologies, Vol. 79. Springer, Singapore, Singapore, pp. 623-636, 2018.

[83] S. Borah, R. Panigrahi, A. Chakraborty, "An Enhanced Intrusion Detection System Based on Clustering," in: K. Saeed, N. Chaki, B. Pati, S. Bakshi, D. Mohapatra (Eds.), Progress in Advanced Computing and Intelligent Engineering, Advances in Intelligent Systems and Computing, Vol. 564, Springer, Singapore, Singapore, pp. 37-45, 2018.

[84] O. Achbarou, M. A. El Kiram, O. Bourkoukou, S. Elbouanani, "A New Distributed Intrusion Detection System Based on Multi-Agent System for Cloud Environment," International Journal of Communication Networks and Information Security (IJCNIS), Vol. 10, No. 3, pp.526-533, 2018.

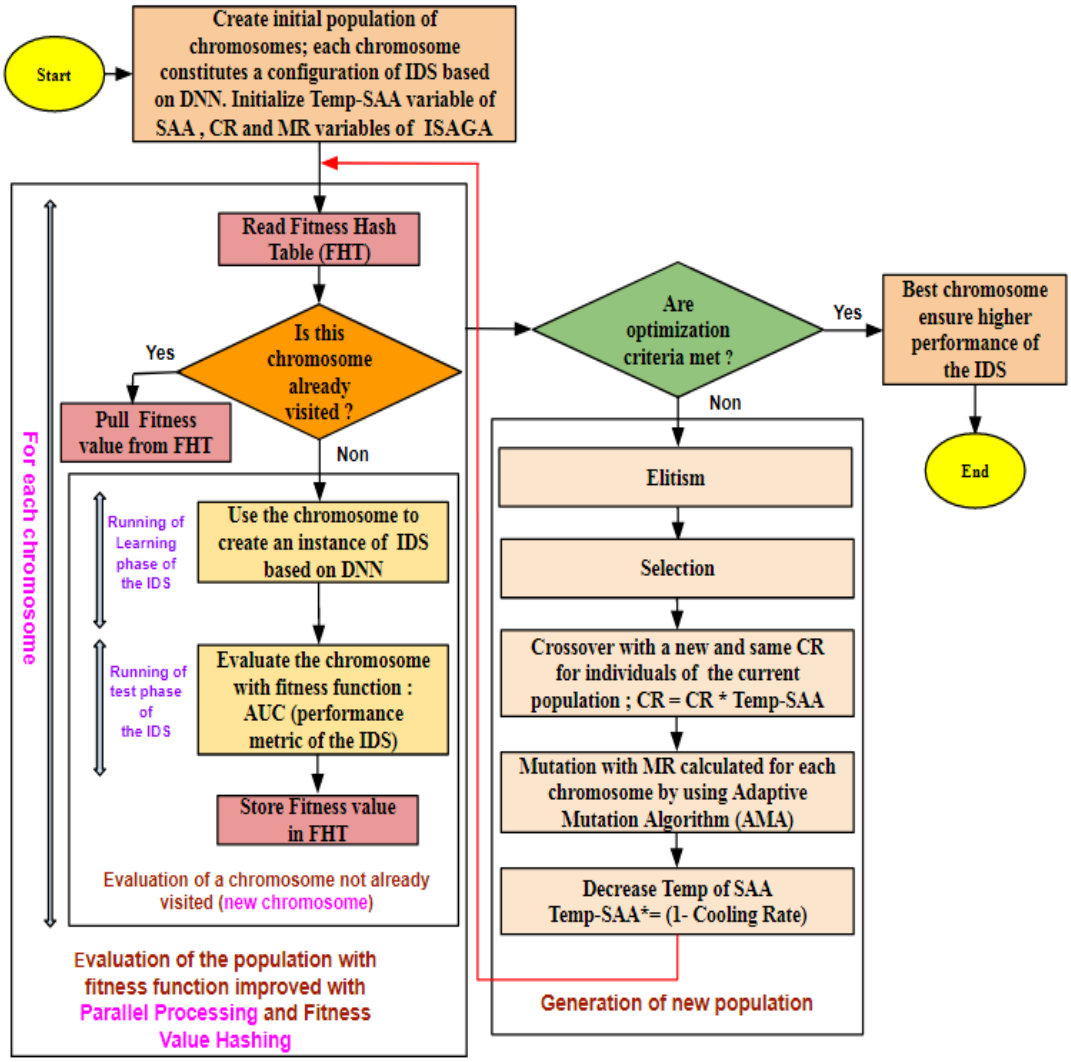

Figure 3. Workflow of proposed system

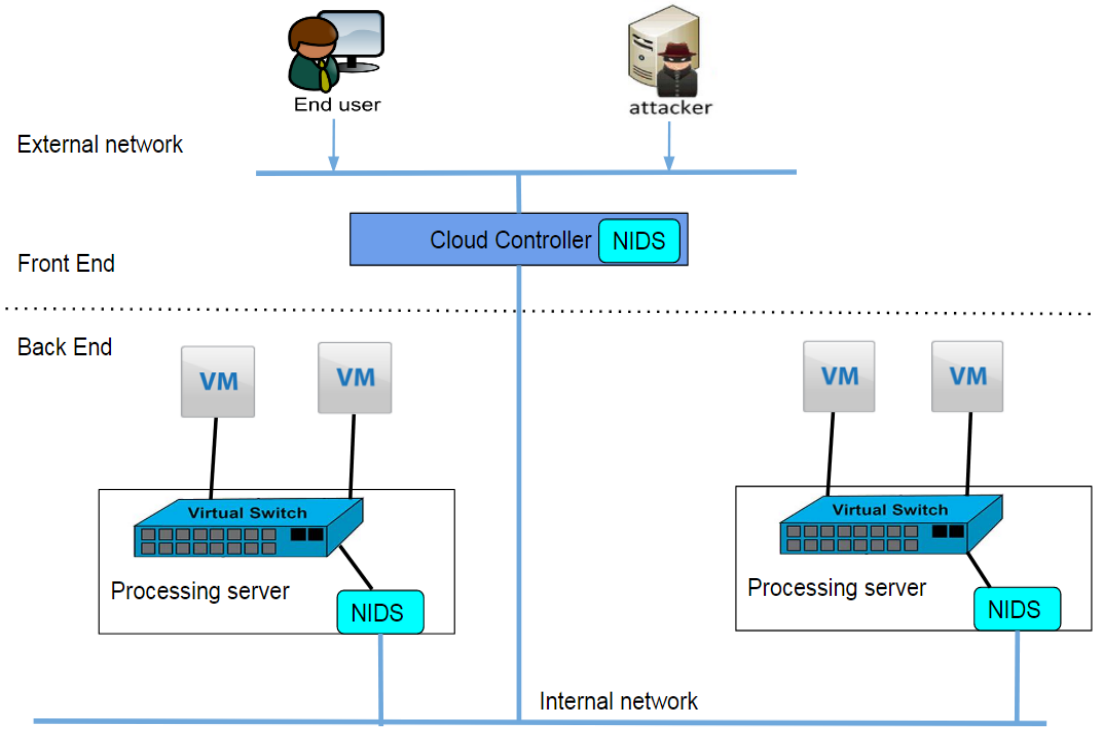

Figure 4. Locations of proposed MLANIDS in a cloud network

Table 14. Description of features used in our study related to Kyoto 2006+ version 2015 dataset [45]

Index feature in Kyoto 2006+ dataset	Name	Description
1	Duration	The length (number of seconds) of the connection
2	Service	The connection's service type, e.g., http, telnet
3	Source bytes	The number of data bytes sent by the source IP address
4	Destination bytes	The number of data bytes sent by the destination IP address
5	Count	The number of connections whose source IP address and destination IP address are the same to those of the current connection in the past two seconds
6	Same_srv_rate	% of connections to the same service in Count feature
7	Serror_rate	% of connections that have "SYN" errors in Count feature
8	Srv_serror_rate	% of connections that have "SYN" errors in Srv_count (the number of connections whose service type is the same to that of the current connection in the past two seconds) feature
9	Dst_host_count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose source IP address is also the same to that of the current connection
10	Dst_host_srv_count	Among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose service type is also the same to that of the current connection
11	Dst_host_same_src_port_rate	% of connections whose source port is the same to that of current connection in Dst_host_count feature
12	Dst_host_serror_rate	% of connections that "SYN" errors in Dst_host_count feature
13	Dst_host_srv_serror_rate	% of connections that "SYN" errors in Dst_host_srv_count feature
14	Flag	The state of the connection at the time the connection was written
18	Label	Indicates whether the session was attack or not; '1' means the session was normal, '-1' means known attack or unknown attack was observed in the session

Table 15. Comparison of performances of our MLANIDS to other works based on shallow or deep neural networks

Research Work	Year	Precision (%)	FPR (%)	Accuracy (%)	DR (%)	F-score	AUC (%)
A deep learning approach for network intrusion detection system [66]	2016	85.44	N/A	88.39	95.95	0.904	N/A
A deep learning approach for intrusion detection using recurrent neural networks [70]	2017	N/A	1.2725	81.29	50.77	N/A	74.74
Comparison of shallow and deep neural networks for network intrusion detection [23]	2018	N/A	1.40	98.50	N/A	N/A	N/A
A deep learning approach to network intrusion detection [71]	2018	99.99	2.15	97.85	97.85	0.9815	97.85
Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection [72]	2018	96.23	N/A	84.96	76.57	0.8528	N/A
A Hybrid Methodologies for Intrusion Detection Based Deep Neural Network with Support Vector Machine and Clustering Technique [73]	2018	N/A	N/A	92.03	91.35	N/A	N/A
Network Intrusion Detection: Based on Deep Hierarchical Network and Original Flow Data [63]	2019	99.84	N/A	99.91	99.92	0.99	N/A
Deep Learning Approach on Network Intrusion Detection System using NSL-KDD Dataset [67]	2019	84.65	19.32	87.17	92.83	0.8855	86.75
Performance Enhancement of Deep Neural Network Using Feature Selection and Preprocessing for Intrusion Detection [24]	2019	N/A	N/A	98.50	N/A	N/A	N/A
Improving the Classification Effectiveness of Intrusion Detection by Using Improved Conditional Variational AutoEncoder and Deep Neural Network [74]	2019	N/A	2.74	85.97	77.43	N/A	87.34
A hybrid machine learning approach for malicious behaviour detection and recognition in cloud computing [30]	2020	96.4	3.6	N/A	96.4	97.5	96.4
An Improved Feedforward Neural Network Using Salp Swarm Optimization Technique for the Design of Intrusion Detection System for Computer Network [31]	2020	N/A	N/A	98.63	N/A	N/A	N/A
Intrusion detection for network based cloud computing by custom RC-NN and optimization [32]	2021	100	0	94.28	80.53	0.8923	90.26
Feedback deer hunting optimization algorithm for intrusion detection in cloud based deep residual network [33]	2021	N/A	0.0544	93.56	93.98	N/A	94.27
Proposed MLANIDS_KYOTO2015	**2021**	**99.99**	**0.02**	**99.95**	**99.95**	**0.99**	**99.96**
Proposed MLANIDS_CIDDS-001	**2021**	**99.47**	**0.04**	**99.96**	**99.95**	**0.99**	**99.95**

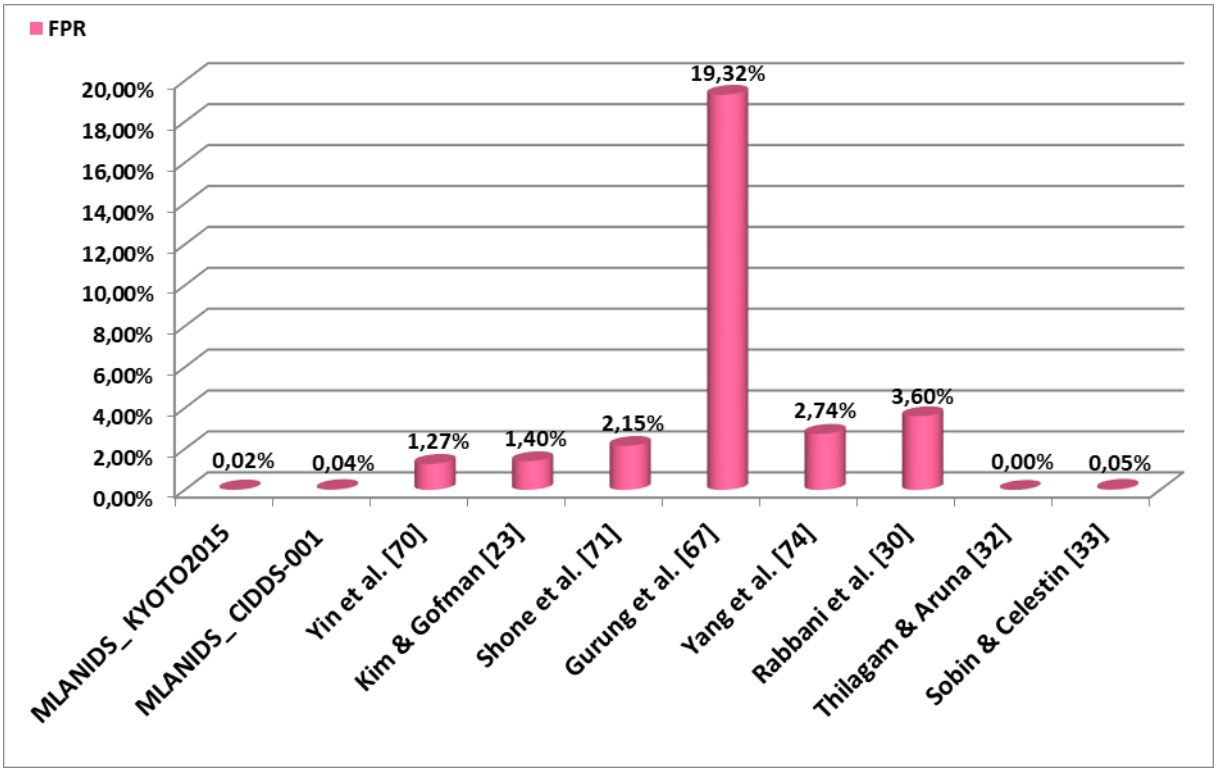

Figure 5. Comparison of False Positive Rate (FPR) of proposed system MLANIDS and other works based ANN

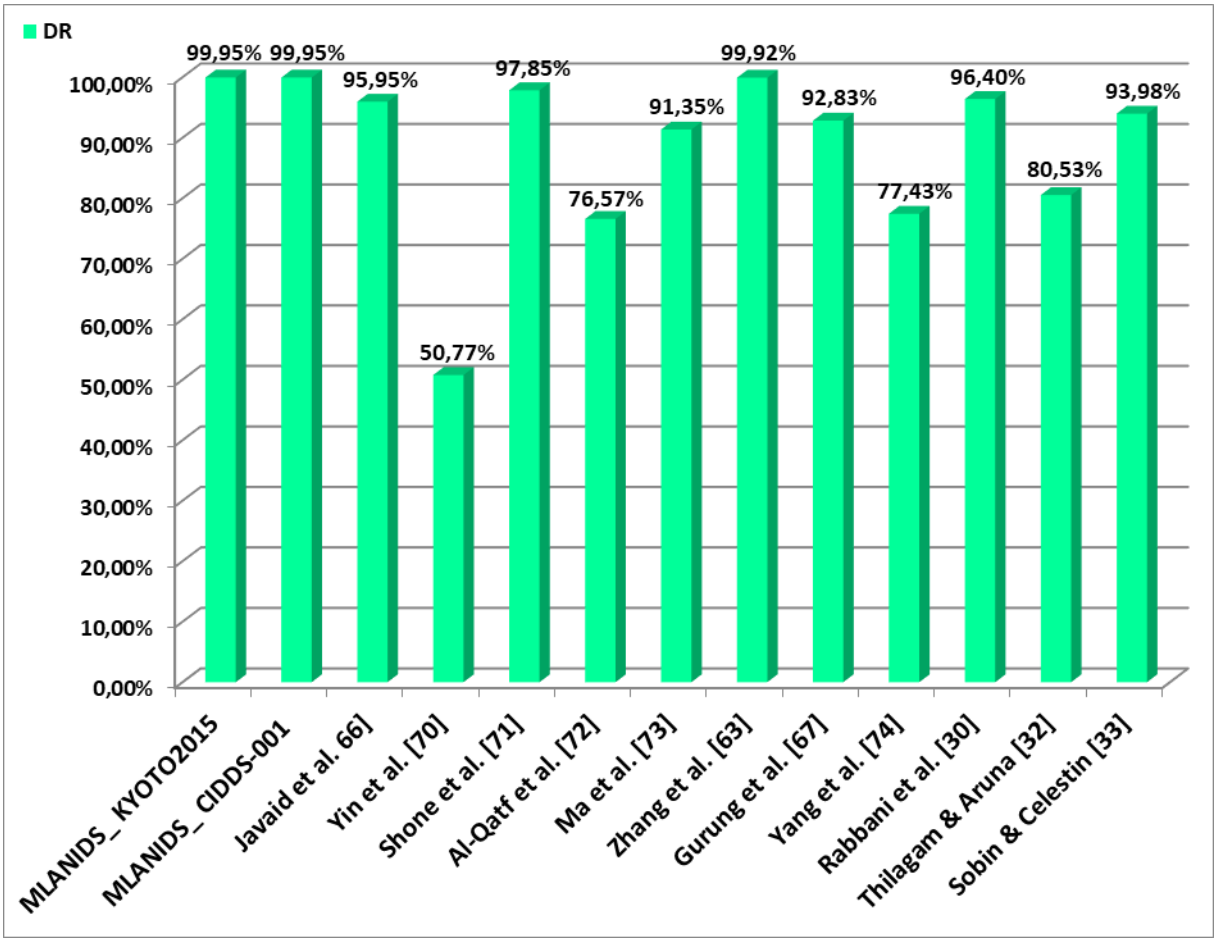

Figure 6. Comparison of Detection rate (DR) of proposed system MLANIDS and other works based ANN

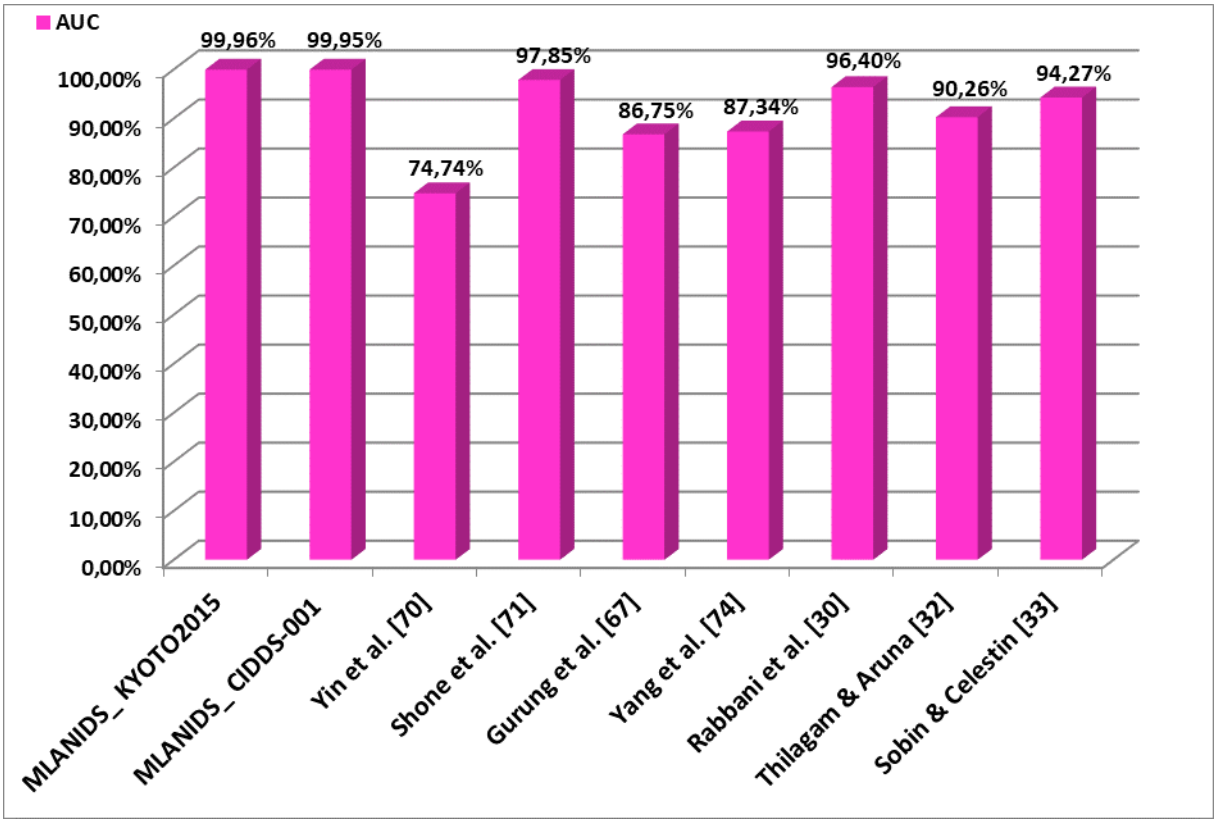

Figure 7. Comparison of AUC (ability to avoid misclassifications) of proposed system MLANIDS and other works based ANN

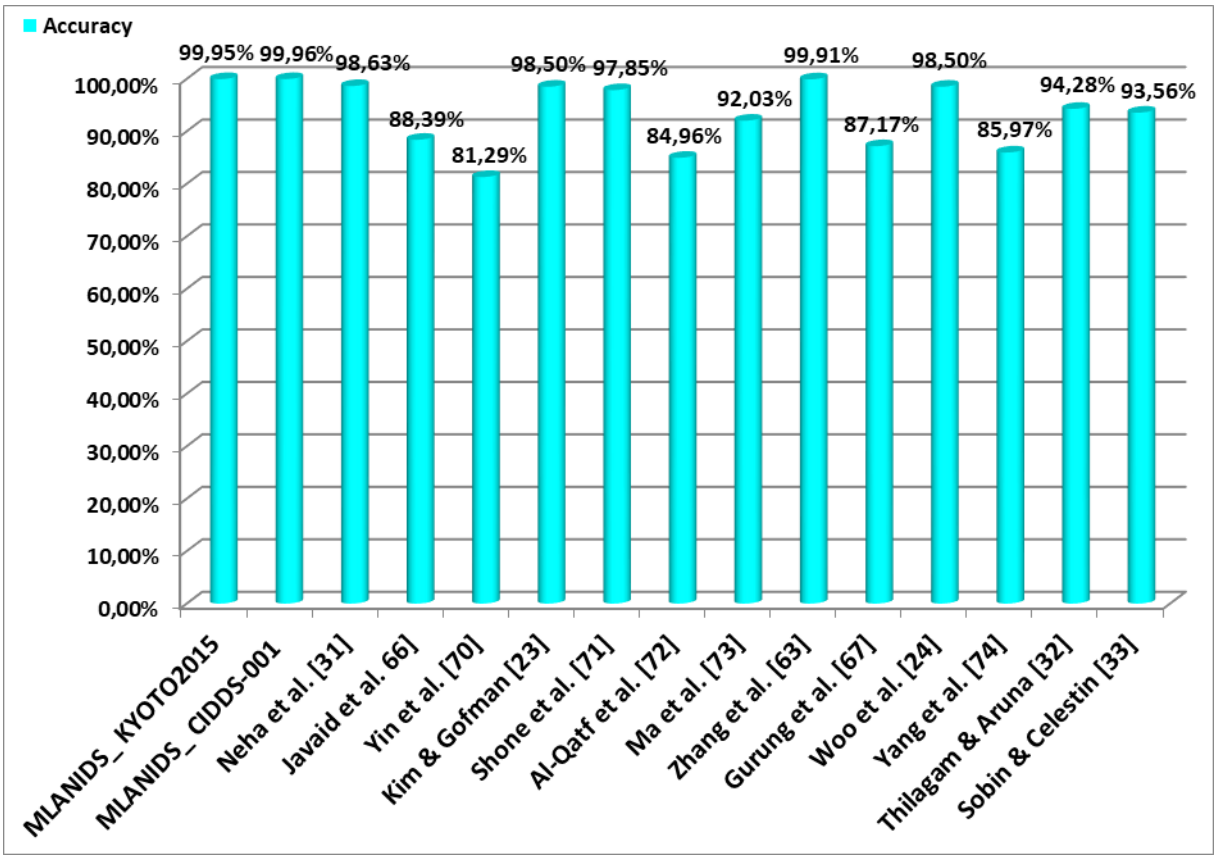

Figure 8. Comparison of Accuracy of proposed system MLANIDS and other works based ANN

Table 16. Comparison of performances of our MLANIDS to other works based on other approaches different to ANN

Research Work	Year	Precision (%)	FPR (%)	Accuracy (%)	DR (%)	F-score	AUC (%)
Distributed intrusion detection system using mobile agents in cloud computing environment [75]	2015	N/A	93	N/A	7	N/A	50
An intrusion detection system using network traffic profiling and online sequential extreme learning machine [76]	2015	N/A	1.74	98.66	99.01	N/A	98.63
Collaborative ids framework for cloud [47]	2016	N/A	1.69	98.92	99.40	N/A	98.85
Intrusion Detection System Based on BCS-GA in Cloud Environment [77]	2016	96.50	3.07	78.23	64.08	77.02	80.50
Another fuzzy anomaly detection system based on ant clustering algorithm [78]	2017	N/A	10.03	N/A	92.11	N/A	91.04
Two-tier network anomaly detection model- a machine learning approach [79]	2017	N/A	4.83	N/A	72.19	N/A	83.68
HIDCC: A hybrid intrusion detection approach in cloud computing [64]	2018	99.12	0.7	99.38	99.38	0.99	99.34
Proposed Network Intrusion Detection System Based on Fuzzy c Mean Algorithm in Cloud Computing Environment [80]	2018	N/A	1.9	99	99	N/A	98.55
Distributed Intrusion Detection System for Cloud Environments based on Data Mining techniques [13]	2018	N/A	N/A	97.05	0.21%	N/A	N/A
Network Anomaly Detection System using Genetic Algorithm and Fuzzy Logic [81]	2018	95.23	0.56	96.53	76.50	0.8484	87.97
An Enhanced Approach to Fuzzy C-means Clustering for Anomaly Detection [82]	2018	99.4567	N/A	95.2992	95.764	N/A	N/A
An Enhanced Intrusion Detection System Based on Clustering [83]	2018	N/A	25	N/A	90	N/A	82.50
A New Distributed Intrusion Detection System Based on Multi-Agent System for Cloud Environment [84]	2018	N/A	12.43	N/A	72.03	N/A	79.80
CS-PSO based Intrusion Detection System in Cloud Environment [12]	2019	96.46	2.87	75.51	59.16	0.73	78.14
Network Intrusion Detection System based PSO-SVM for Cloud Computing [4]	2019	99.50	0.87	99.10	99.08	0.99	99.10
BNID: A Behavior-based Network Intrusion Detection at Network-Layer in Cloud Environment [25]	2019	N/A	1.517	98.888	99.182	N/A	98.84
Cloud Security: LKM and Optimal Fuzzy System for Intrusion Detection in Cloud Environment [26]	2019	96.54	N/A	N/A	89.26	91. 83	N/A
A Distributed Intrusion Detection Scheme for Cloud Computing [27]	2020	N/A	0.03	99.6	99.7	N/A	N/A
Anomaly-Based Intrusion Detection System Using Support Vector Machine [30]	2020	N/A	N/A	96.34	N/A	N/A	N/A
Building an Ensemble Learning Based Algorithm for Improving Intrusion Detection System [31]	2020	N/A	N/A	95	N/A	N/A	N/A
Proposed MLANIDS_KYOTO2015	**2021**	**99.99**	**0.02**	**99.95**	**99.95**	**0.99**	**99.96**
Proposed MLANIDS_CIDDS-001	**2021**	**99.47**	**0.04**	**99.96**	**99.95**	**0.99**	**99.95**