Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

DL-IDS: A Deep Learning-based Intrusion Detection Framework for Securing IoT

Yazan Otoum*[1] | Dandan Liu[2] | Amiya Nayak[1]

[1]School of Electrical Engineering & Computer Science (SITE), University of Ottawa, ON, Canada
[2]School of Computer Science , Wuhan University, Wuhan, China

Correspondence
*Yazan Otoum, School of Electrical Engineering & Computer Science, University of Ottawa, ON, Canada, Email: yazan.otoum@uottawa.ca

Abstract

The Internet of Things (IoT) is comprised of numerous devices connected by wired or wireless networks, including sensors and actuators. Recently, the number of IoT applications, including Smart Homes, VANET's, Health-Care, Smart Cities and Wearables, has increased dramatically. The number of connected devices is projected to jump from approximately 27 billion in 2017 to 125 billion in 2030, with an average annual increment of 12% as reported in IHS Markit[1]. Currently, security is a critical issue in the IoT field due to the nature of its architecture and the types of devices, as well as different methods of communication (mainly wireless) and the volume of data being transmitted in the network. Security becomes more important as the number of devices connected to the IoT increases. To overcome the challenges of securing IoT devices, we propose a new Deep Learning-based Intrusion Detection System (DL-IDS) to detect security threats in IoT environments. Many IDSs can be found in the literature, but they lack optimal feature learning and dataset management, significant issues that affect the accuracy of attack detection. Our proposed module combines the Spider Monkey Optimization algorithm (SMO) and the Stacked-Deep Polynomial Network (SDPN) to achieve higher detection accuracy; SMO selects the optimal features in the datasets, and SDPN classifies the data as normal or anomaly data. Types of anomalies detected by DL-IDS include Denial of Service (DoS), User to Root (U2R) attack, probe attack and Remote to Local (R2L) attack. Extensive analysis shows that the proposed DL-IDS achieves better performance in terms of accuracy, precision, recall and F-score.

KEYWORDS:
Internet of things (IoT) Security, Deep Learning (DL), Intrusion Detection System (IDS), Stacked-Deep Polynomial Network (SDPN), Spider Monkey Optimization (SMO).

1 | INTRODUCTION

Recently, the number of IoT applications has increased dramatically. IoT is made up of a large number of different physical endpoints or sensors. These are connected to one another and the Internet, and they can collect data from the surrounding environments and share it with each other. Small devices with low power and limited resources can also send data to the IoT gateways, where it is aggregated and connected to the endpoint networks on the Internet. Security in IoT field is becoming a more and more challenging issue along with the IoT industry growing. It is mainly due to the heterogeneity of IoT architecture,

the different types of accessed devices and multiple approaches of communication[2–4], as well as the huge volume of data being transmitted through the network. As shown in Fig. 1 IoT attacks have increased during the last year 2018 by 217.5%, from 10.3 million in 2017 to 32.7 million, according to 2019 Cyber Threat Report logged by SonicWall[5] .

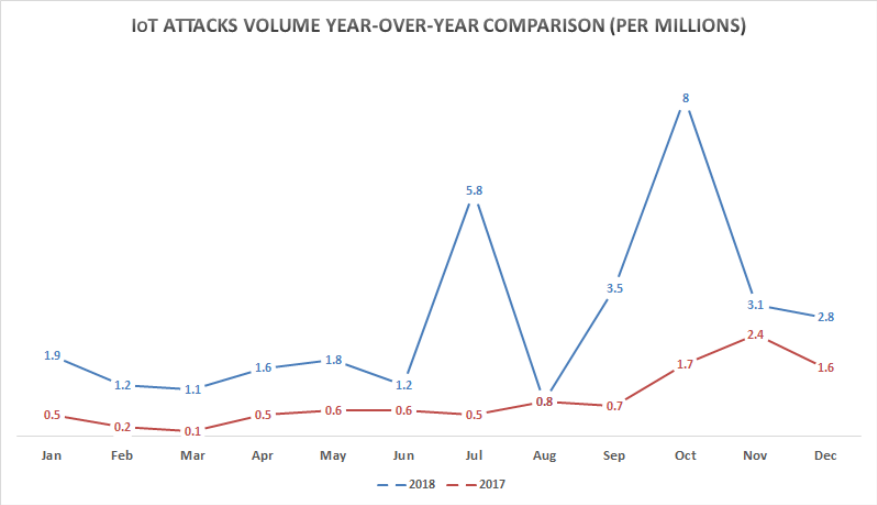

FIGURE 1 IoT attacks volume year-over-year comparison[5]

Security issue usually involves authentication, data privacy, availability, confidentiality, integrity, energy efficiency, single-point failures to be verified, and so on[6,7]. Most security threats can be categorized as:

- High-level threats: insecure interfaces, insecure software/firmware, and middle-ware security;
- Intermediate level threats: routing disruptions, replay attacks, insecure neighbour discovery, buffer reservation, sinkholes, authentication, session establishment, and privacy violations; and,
- Low-level threats: jamming, Sybil, spoofing, insecure initialization, and sleep deprivation attacks.

In dealing with the security problem in IoT, many Intrusion Detection Systems (IDSs) can be found in the literature. In traditional IDS, data are collected by sensors and sent to the analysis engine, which is responsible for examining the collected data and detecting intrusions. If an intrusion is detected, the reporting system generates an alert to the network administrator. IDS approaches can be further classified into signature-based and anomaly-based methods, according to whether the system or network behaviours match with an attack signature or the deviation from the normal ones exceeds the threshold. However, since traditional methods are usually lack of optimal feature learning and dataset management, the accuracy of attack detection can not be guaranteed since dealing with irrelevant features in the high dimensional data generated from thousands of IoT sensors and devices can be increase the risk of over-fitting through the opportunity to make decisions based on the noise. and extends the training time. In this proposed model we have shown in section 5 that how SMO has provide these benefits to our proposed system by choosing the optimal features. Machine Learning (ML) and Deep Learning (DL) approaches are recently proposed[8,9,10] for detecting and mitigating security threats. For attack detection, ML algorithms such as Support Vector Machines (SVM), K-Nearest Neighbour's algorithm (KNN), Decision Trees (DT), Bayesian algorithms, Random Forest and K-means algorithm are typically applied in IDS. Similarly, DL approaches such as Deep Belief Network (DBN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Restricted Boltzmann Machine (RBM) and deep AutoEncoder (AE) are also employed in IDSs. However, the DL approach has achieved better performance results over ML, especially in large datasets. Several IoT systems produce a substantial volume of data, while DL methods are suitable for such a system. It provides the support of high-dimensional features. It also enables the deep linking which permits IoT-based devices and their applications to interact with one another automatically without human intervention. Though security and privacy are the primary goals of DL IoT data processing, it has also been widely used for image processing, Big Data analytics, video processing, and speech processing in IoT and smart city applications[11].

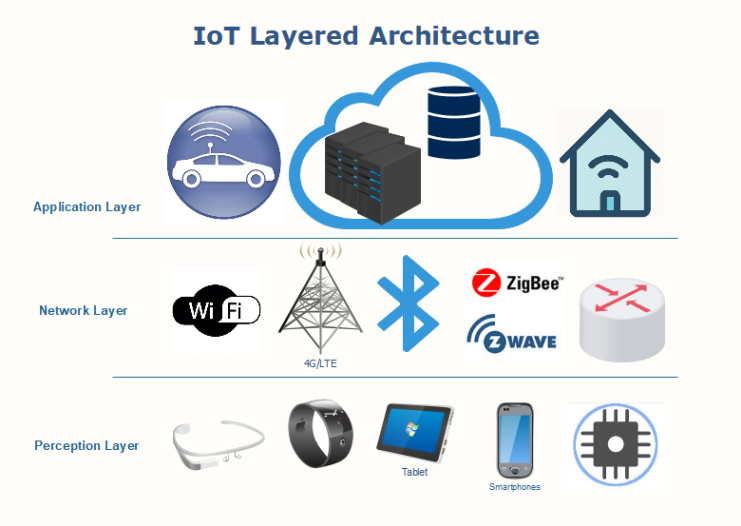

FIGURE 2 IoT Architecture

Therefore, to overcome the challenges of protecting IoT, we propose a novel deep learning-based framework for IoT environments. The main contributions of this paper are as follow:

- A novel Deep Learning-based IDS (DL-IDS) design to secure IoT environments with improved efficiency of attack detection;
- A model that can handle datasets with uncertain or missing data, and redundant values;
- An SMO algorithm that is improved upon other optimization algorithms in terms of convergence time is proposed for feature learning. The optimal packet features are identified by the SMO algorithm and extracted from the cleaned dataset;
- Based on the learned optimal features, SDPN classifies the data as normal or anomalous (DoS, U2R, probe and R2L). Dataset cleaning and optimal feature learning help SDPN achieve higher accuracy and shorter training time, compared with other methods.

The rest of this paper is organized as follows: Section 2 provides a background of IoT security and deep learning, while the state-of-the-art research done in the IoT security using Deep learning are discussed in Section 3. Section 4 explains the proposed DL-IDS with the algorithms, and Section 5 evaluates the performance of DL-IDS in terms of performance metrics. In the last section, we conclude our contributions and highlight potential future research in DL-IDS IoT security.

2 | BACKGROUND

2.1 | IoT Security

IoT architecture consists of three layers (Fig. 2), known as the application, network and perception layer [12]. The application layer is the top level in the IoT architecture and provides services to users such as smart cities and smart grids. The network layer is the most crucial in IoT architecture, as it integrates the connecting devices used in the IoT infrastructure, including gateways and switches, with different communication technologies and protocols such as Wi-Fi, ZigBee and Bluetooth. The perception layer, also known as the sensor layer, collects and processes the data from sensors, or controls physical objects in the near environment such as actuators . The perception layer has two sections: sensor nodes like RFID tags, and wireless sensor networks, which are self-organizing and contain many sensor nodes deployed in large areas.

The IoT Gateway plays a crucial role in IoT systems since it is responsible for data forwarding, controlling nodes and communication protocol compatibility. It has a wide range of access capabilities, including seamless manageability and protocol interworking between traditional network and WSN. Manageability Protocol interworking support protocol interworking

between the traditional network and WSN seamlessly. Each layer in an IoT system is susceptible to threats and should be secured through one or more techniques of effective anomaly detection, such as authentication and encryption [13]. Table 1 indicates the IoT security threats at each layer.

TABLE 1 Attack Classifications

Layer	*Security threats*	*Threats description*
Perception Layer	Node capture attacks, Malicious code injection attacks, false data injection attacks, Replay attacks, Cryptanalysis attacks and side channel attacks, eavesdropping and interference, Sleep deprivation attacks.	The security threats in the perception layer is focusing on forging the aggregated data from the IoT devices and destroy the perception devices. [12]
Networks Layer	DoS attacks, Spoofing attacks, Sinkhole attacks, Wormhole attacks, Man in the middle attack, Routing information attacks, Sybil attacks, Unauthorized access.	The security threats in this layer is focusing on effecting the availability of the network resources since the main purpose of the network layer in the IoT environments is to transmit the aggregated data.However, most of the devices in the IoT environment are connecting via wireless communications. [12]
Application Layer	Phishing attack, Malicious virus/worm, Malicious scripts.	The security threats in the application layer is focusing on the software attacks since the main purpose of this layer is to support the services requested by the users. [12]

We describe some of the network layer threats and attacks on the IoT systems. DoS attack is the most common attack in the IoT networks. It makes the network services unavailable to its authorized users by utilizing all of its resources through hits the network with a massive amount of traffic [14]. In a man in the middle attack, a malicious device controlled by the attacker can intercept the communications between two authorized devices and play as a store and forward device in aims to modifying, eavesdropping, and controlling the communication between them. In a spoofing attack, the attackers can spoof the address of the IoT valid device in aim to make communication with the other network devices and send a malicious data as a valid device.

2.2 | Deep learning for IoT

Deep learning (DL) is a state-of-the-art machine learning (ML) field with powerful analytics capabilities, used for applications such as computer vision, bioinformatics and natural language processing. It is seen to demonstrate significant performance improvement over some ML algorithms and has recently been used in some IoT applications in edge/fog computing. These applications require substantial volumes of data to be transferred over the network, and DL gives better performance with extensive data than ML. Besides, DL can work with new features that can be used to solve the problems without human interactions [15].

3 | LITERATURE SURVEY ON DL APPROACHES

DL has been used for many IoT applications for the reasons discussed above. Lane et al. [16] used two of the most common deep learning algorithms (i.e. Convolutional Neural Networks (CNN) and Deep Neural Networks (DNN) to build a model that processes sensor data using three different types of IoT hardware employed in wearables and smartphones. DL approach-based authentication using Long Short Term Memory (LSTM) that learns the hardware imperfections of low-power ratios has also been proposed for IoT environments [17]. LSTM was designed to be sensitive to signal imperfections in device identification. However, the method is only helpful for device identification as it cannot detect other significant attacks. LSTM has also been applied for botnet detection in IoT [18]. In addition, an LSTM-based IoT threat hunting approach involving OpCode extraction, feature

extraction and deep threat classification was proposed. Though optimal feature selection based on term frequency and inverse document frequency was also tested (TF-IDF), it is only usable with small datasets and is unsuitable for real-time applications. In [19], a Deep Neural Network (DNN) was used for attack classifications in IoT environments, and its performance was evaluated by cross-validation and sub-sampling. A grid search method was used to initialize the parameters of DNN; the parameter learning using grid search takes a long time to initialize optimal parameters. The learning-based deep-Q-network was proposed to maintain security and privacy in IoT healthcare environments [20] by managing authentication, access control and intermediate attacks in IoT. However, attack detection based on non-optimal features leads to degradation of classification accuracy. Botnet detection on the internet of battlefield things (IoBT) was performed using the deep Eigenspace learning approach [21]. The operation code (OpCode) sequence of devices was used for malware detection before classification, and a feature-based graph was constructed for each sample. As the involvement of large computations results in high computational complexity, this is ineffective for IDS. For botnet detection, a bidirectional LSTM-RNN method was used in IoT [22], and word embedding was applied for text recognition and to convert the attack packet into integer format. However, this method only performs well with a limited number of attack vectors; when the number increases, it becomes ineffective. Stacked AutoEncoder (AE) is an unsupervised DL method proposed to detect intrusion in fog enabled IoT environments [23], and its attack detection latency and scalability were improved by using fog computing in IoT. However, the running time of AE is relatively long, and the detection accuracy is low due to non-optimal features.

This literature survey reveals the following research challenges:

- Dealing with uncertain data limits the accuracy of DL modules. However, there are huge volumes of uncertain data in real-time environments.
- Processing irrelevant features (non-optimal) degrades the accuracy of IDS and extends training time.
- A deep learning approach that can overcome all the above issues in LSTM and DBN is required.

All these problems are resolved by our proposed DL-IDS, which combines optimization algorithms and DL approaches.

4 | PROPOSED DL-IDS

This section provides detailed explanations about using DL-IDS to secure IoT environments. The overall process of DL-IDS depicted in Fig. 3, shows that it starts with preprocessing to remove uncertainties in which the dataset is normalized. Preprocessing applies two effective processes: redundancy elimination and missing value replacement. The cleaned dataset is then processed with an optimal feature selection algorithm to extract relevant features. Based on the optimal features, the data is classified into normal and anomalous data.

Here, we consider four categories of anomalies (DoS, Probing, U2R and R2L) which can be defined as follows:

1. DoS: An attacker tries to make a service unavailable to legitimate users by uploading enormous unwanted packets. DoS attacks include Apache2, Back, Land, Udpstorm and Smurf.
2. Probing: In this type of attack, an attacker monitors the remote victim and tries to collect some information by doing the port scanning. Attacks such as duration of the connection and source bytes are some types of Probing attack.
3. R2L: In this type of attack, an unauthorized attacker attempts to access the system without a system account. R2L attacks include Ftpwrite, Guesspasswd and SNMP.
4. U2R: In this scenario, the attacker has local access to a victim's machine and tries to obtain legitimate user privileges. Buffer overflow, HTTP tunnel and rootkit attacks are types of U2R attacks.

4.1 | Preprocessing Phase

This process is initialized with similarity measurements of the data in the dataset, using the Minkowski distance to compute the distance between each pair of data. Duplicate and redundant data are then removed from the dataset and fed into the next stage, where the missing values are replaced by nearest neighbour computation to avoid the classifier to biased towards more

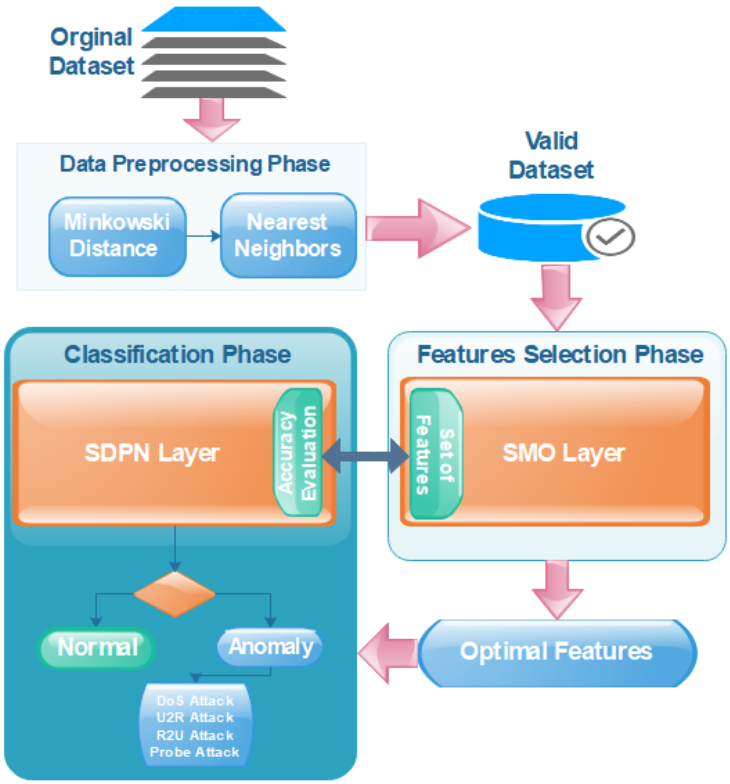

FIGURE 3 The Proposed DL-IDS Framework

frequent records. Missing value replacement, or imputation, is the process of predicting the relevant value of missed attributes in the data. This missing value replacement method identifies the nearest neighbours of missing values, and the mean value of the missing attributes is replaced in the missing values. Consider a dataset with m records in which each record has n attributes, and x attributes are missing in the data record R_1. Then, the K nearest neighbours of R_1 are determined according to the Euclidean distance (ED) as follows:

$$ED = \sum_{i=2}^{m} |R_1 - R_i|^2 \tag{1}$$

Based on the distance value, the K nearest neighbour records are identified for R_1, and the attribute x value of R_1 is computed from the attribute value of its K neighbours. The mean value of x is computed for the x values presented in neighbour records, and the missing value is replaced by this mean value. This step eliminates all redundant data, and the missing values are replaced to remove all uncertainties from the dataset.

4.2 | Optimal Feature Selection Phase

In the preprocessing step, all uncertainties were removed from the dataset, and in this step, we select the optimal features from the dataset. Optimal feature selection minimizes the feature learning time in our SDPN classifier. Here, we adapt the SMO algorithm for optimal feature selection[24]. SMO is a food foraging-based optimization algorithm inspired by fission-fusion social systems. It has been used in many engineering domains due to the few required number of control parameters which makes it easy to apply in solving complex optimization problems.following are the main control parameters with it's values:

- LocalLeader Limit = [S X D]
- GlobalLeader Limit = [S/2,SX2]
- Maximum no. of groups MG = S/10

- Perturbation rate PR = [0.1,0.9]

Where, S is the population size, and D is the dimension. Algorithm 1 explains the process of optimal feature selection in the SMO algorithm.

Algorithm 1 SMO based feature selection

Begin
Set value of control parameters
Initialize swarm with initial population of SMOs
Compute *fitness* for all SMO in the swarm
Select global leader and local leader
Set iteration = 0

while (Iteration < Maximum iterations) **do**
//Local Leader Phase
Compute the probability of a particular solution
Update the position of each SMO in the group
//Global Leader Phase
Update the position of each SMO in the swarm
//Global Leader Learning phase
Update the position of the global leader
Select global leader with highest *fitness*
//Local Leader Learning phase
Update position of the local leader
Select local leader based on *fitness*
Iteration = Iteration+1;

End

Following are the details of SMO steps; ***Initialization:*** In this step, all solutions are initialized as spider monkeys. In our work, the 41 features are initialized as a population of N spider monkeys where each monkey $SM_i (i = 1, 2, ..., N)$ is a D-dimensional vector, and SM_i indicates the i^{th} Spider Monkey (SM) in this population. Each dimension j of SM_i is initialized as follows:

$$SM_{ij} = SM_{minj} + Rand\,[0, 1]\,(SM_{maxj} - SM_{minj}) \tag{2}$$

where, SM_{minj} and SM_{maxj} are limits of SM_{ij} in j^{th} direction, and $rand[0, 1]$ is a random value in the range [0-1].
Local Leader Phase (LLP): In this stage, each SM modernizes its present location, based on the experience of each local leader and local group members. The fitness value of so acquired new position is defined. In case the fitness value of the new position is better than the current (higher is better), then the spider monkey updates its position with the new one, as follows:

$$SM_{newij} = SM_{ij} + rand\,[0, 1]\left(LL_{kj} - SM_{ij}\right) + rand[-1, 1](SM_{rj} - SM_{ij}) \tag{3}$$

where the j^{th} dimension of the i^{th} SM is indicated by SM_{ij}, the j^{th} dimension of the k^{th} local group leader position is denoted by LL_{kj}. SM_{rj} indicates the j^{th} dimension of the r^{th} SM which is selected randomly from the K^{th} local group where $r \neq i$.
Global Leader Phase (GLP): After finishing the local leader phase, this phase can start where all spider monkeys update their positions based on the experience of the global leader and local group members as follows:

$$SM_{newij} = SM_{ij} + rand\,[0, 1]\left(GL_j - SM_{ij}\right) + rand[-1, 1](SM_{rj} - SM_{ij}) \tag{4}$$

where, GL_j is the j^{th} dimension of the GLP and $j \in \{1, 2, \ldots, D\}$ is a randomly selected index. The positions of spider monkeys (SM_i) are updated based on the probability $Prob_i$ which are defined utilizing their fitness. In this way, the best candidate will have higher possibility to be the leader in the next phase. The probability $Prob_i$ is calculated as follows (or by any other function of fitness):

$$Prob_i = \frac{Fitness_i}{\sum_{i=1}^{N} Fitness_i} \tag{5}$$

TABLE 2 Selected Features by SMO

No.	Selected By SMO	Type	Feature	No.	Selected By SMO	Type	Feature
1		Continuous	duration	23	√	Continuous	count
2	√	Symbolic	protocol_type	24		Continuous	srv_count
3	√	Symbolic	service	25	√	Continuous	serror_rate
4	√	Symbolic	flag	26		Continuous	srv_serror_rate
5	√	Continuous	src_bytes	27		Continuous	rerror_rate
6	√	Continuous	dst_bytes	28		Continuous	srv_rerror_rate
7	√	Symbolic	land	29	√	Continuous	same_srv_rate
8	√	Continuous	wrong_fragment	30	√	Continuous	diff_srv_rate
9		Continuous	urgent	31		Continuous	srv_diff_host_rate
10	√	Continuous	hot	32		Continuous	dst_host_count
11		Continuous	num_failed_logins	33		Continuous	dst_host_srv_count
12	√	Symbolic	logged_in	34		Continuous	dst_host_same_srv_rate
13		Continuous	num_compromised	35	√	Continuous	dst_host_diff_srv_rate
14		Continuous	root_shell	36	√	Continuous	dst_host_same_src_port_rate
15		Continuous	su_attempted	37	√	Continuous	dst_host_srv_diff_host_rate
16		Continuous	num_root	38	√	Continuous	dst_host_serror_rate
17		Continuous	num_file_creations	39	√	Continuous	dst_host_srv_serror_rate
18		Continuous	num_shells	40	√	Continuous	dst_host_rerror_rate
19		Continuous	num_access_files	41		Continuous	dst_host_srv_rerror_rate
20		Continuous	num_outbound_cmds				
21		Symbolic	is_host_login				
22		Symbolic	is_guest_login				

where $Fitness_i$ refers to the fitness value of the i^{th} SM. In this way, the fitness of the last position is defined and compared with the old one, and then the best is selected. In this work, the evaluation of each feature (spider monkey) is executed by SDPN classifier to calculate the efficiency of signified feature subset (which is here the accuracy).

Global Leader Learning phase (GLL): In this phase, the global leader modifies its location by applying a greedy global selection to all the population set. Then the SM with the highest fitness value becomes the global leader. A counter called *GlobalLimit-Count* is incremented by 1 in case of the global leader position does not update.

Local Leader Learning phase (LLL): In this phase, the local leader in each group updates its position by applying the greedy selection in the same group, and the SM with the highest fitness value in each group becomes the local leader. A counter called *LocalLimitCount* is incremented by 1 in case of the local leader position does not update.

Local Leader Decision (LLD): In this phase, if the Local Limit Count reaches its threshold value, then the group members changes their positions by using gathered information from Global and Local Leaders as the equation:

$$SM_{newij} = SM_{ij} + U(0,1) * (GL_j - SM_{ij}) + U(0,1) * (SM_{ij} - LL_{kj}) \tag{6}$$

Global Leader Decision (GLD): In this phase, if the global leader does not reach the maximum number of iterations (Global Leader Limit), Then the global leader divides the population into smaller local groups until reach the maximum number of groups (MG). In this case, if the position of global leader is not updated, then it combines all the groups in one group.

Moreover, the final best subset of features with high classification accuracy is defined as the better optimal result. Therefore, this algorithm simulates the fusion-fission social system of SMs. Thus, in this process, the optimal feature set is selected by the SMO algorithm. Table 2 shows the selected NSL-KDD features by SMO that gives higher accuracy.

4.3 | Classification Phase

In the classification phase, we adopted SDPN which has been used with numerous small and large datasets and has shown high-performance results in [25,26]. In SDPN, multiple basic DPNs are stacked up to form a deep hierarchy. DPN a new supervised deep neural network algorithm with efficient layer-by-layer learning in which the output of each node is a quadratic function of its inputs. The highest level output representation is then fed to an SDPN classifier. The elimination of irrelevant features by selecting optimal features minimizes the SDPN training time. Moreover, the optimal means of building deep networks in SDPN allows data representation learning on small finite samples. The algorithm starts with a simple network that could have significant bias but will not tend to over-fit (i.e. low variance), and as the network becomes deeper, we can gradually decrease the bias while increasing the variance. Thus, this algorithm could be applied to train the natural curve of the solutions used to control the bias-variance trade-off.

The construction of the layers in DPN begins by assigning the degree-1 polynomial function (linear) over the training dataset, where the bias is built using Singular Vector Decomposition (SVD) to generate an independent set of vectors. The outputs of the first layer extend all values obtained by the linear functions on the training instances. Then, by using the same concept, the basis for degree-2 and three polynomials can be derived. This means that the vector of values achieved by any degree-2 polynomial extends the vector of values obtained by nodes in the first layer and products of the outputs of every two nodes in the first layer.

Letting $R= \{r_1,r_2, \dots ,r_m\} \in R^{mxd}$ gives a set of m training samples, where r_i is a d-dimensional sample. Then, the construction of the first layer in DPN begins by assigning the degree-1 polynomial function (linear) over the training data as follows:

$$\{ (\langle w \ , \ \left[1 \ r_1\right] \rangle \ , \dots , \langle w \ , \ \left[1 \ r_m\right] \rangle) : \ w \in R^{d+1} \} \tag{7}$$

which is the (d+1)-dimensional linear subspace of R_m. Therefore, to build the basis we need to find d+1 vectors $w1, \dots , w_{d+1}$. So, $\{ (\langle w_j \ , \ \left[1 \ r_1\right] \rangle \ , \dots , \langle w_j \ , \ \left[1 \ r_m\right] \rangle) \}_{j=1}^{d+1}$ are linearly independent vectors ,this can be generated by using Singular Vector Decomposition (SVD) to end up with linear transformation that specified by a matrix W, which maps [1 X] into the generated basis, in which the columns of W specify the d+1 linear function to form the first layer of the network. The j^{th} node of the first layer functions as:

$$n_j^1 (r) \ = \ \langle W_j \ , \ [1 \ R] \rangle \tag{8}$$

where $\{ (n_j^1(r_1) \ , \dots , \ n_j^1(r_m)) \}_{j=1}^{d+1}$ is the basis for all values obtained by degree-1 polynomials over the training dataset. If F^1 refers to $m \times (d+1)$ matrix which columns are the vectors of this matrix, then $F_{i.j}^1 = n_j^1(r_i)$.
In this way, the network of the first layer is constructed, whose outputs extend all the values obtained by linear functions on the training dataset. In the same way, basis of degree-2 and 3 can be found. Then any degree s polynomial can be expressed as:

$$\sum_{\mathrm{i}} \mathrm{g_i(r)\ h_i(r)+k(r)} \tag{9}$$

where, $g_i(r)$ are degree-1 polynomials, $h_i(r)$ are (s-1)-degree polynomials, and k (r) is polynomial of degree at most s-1. The nodes in layer-1 extend all degree-1 polynomials then it extends the polynomials g_i , h_i ,and k , So we can express any degree-2 polynomial as:

$$\sum_i \left(\sum_j \alpha_j^{(g_i)} n_j^1(r) \right) \left(\sum_x \alpha_x^{(h_i)} n_x^1(r) \right) + \left(\sum_j \alpha_j^{(k)} n_j^1(r) \right) = \left(\sum_{j,x} n_j^1(r) n_x^1(r) \right) \left(\sum_i \alpha_j^{(g_i)} \alpha_x^{(h_i)} \right) + \sum_j n_j^1(r) (\alpha_j^{(k)}) \tag{10}$$

where α's are scalar factors. This means that the vector of values achieved by any degree-2 polynomial extend the vector of values obtained by nodes in the first layer, and products of the outputs of every two nodes in the first layer. For the algebraic representation, when the first layer was constructed a matrix F^1 was formed, which columns extend all values obtained by degree-1 polynomials. Then the matrix $[\mathrm{F}\ \widetilde{F}^2]$ can be expressed as:

$$\widetilde{F}^2 = [(F_1^{1\circ} \ F_1^1) \dots (F_1^{1\circ} \ F_{|F_1|}^1) \ \dots (F_{|F^1|}^{1\circ} \ F_1^1) \dots (F_{|F^1|}^{1\circ} \ F_{|F1|}^1)] \tag{11}$$

where, $\circ$ refers to the direct matrix product. We can define F^2 as the method as F^1, then the new matrix$[F \ \widetilde{F}^2]$ extends all possible values of degree-2 polynomials, then $[\mathrm{F\ F^2}]$'s columns are linearly independent basis for $[F \ \widetilde{F}^2]$'s columns.

Now, to construct layer 3, we need only to repeat the previous process, at each iteration s, matrix F is maintained, whose columns form a basis for the values obtained by all polynomials of degree $\leq s-1$. then we can write the new matrix as:

$$\widetilde{F}^{s} = [(F_1^{s-1} \circ F_1^1) \ldots (F_1^{s-1} \circ F_{|F_1|}^1) \ldots (F_{|F^{s-1}|}^{s-1} \circ F_1^1) \ldots (F_{|F^{s-1}|}^{s-1} \circ F_{|F1|}^1)] \tag{12}$$

After some $\Delta - 1$ iterations a matrix F will be constructed, whose columns form a basis for all values obtained by polynomials of degree $\leq \Delta - 1$ over the training dataset. Algorithm 2 summarizes the steps for building DPN.

Algorithm 2 Main algorithm flowchart to build the networks in DPN

Begin

Initialize an empty matrix (F) , and $\widetilde{F}^{1} := [1 \;\; X]$

Find W by computing SVD of $\widetilde{F}^{1}$

Build basis for layer 1: $(F^{1}, W^{1}) := (\widetilde{F}^{1})$

// Columns of F^1 are linearly independent, and $F^1 := \widetilde{F}^{1} W^1$

Building first-layer: $\forall i \in \{1, \ldots, |F^1|\}, n_i^1(r) := \langle W_i, [1 \; R] \rangle$

$F := F^1$

Build higher layers s = 2 and 3:

$\widetilde{F}^{s} := [(F_1^{s-1} \circ F_1^1) \ldots (F_1^{s-1} \circ F_2^1) \ldots (F_{|F^{s-1}|}^{s-1} \circ F_{|F^1|}^1)]$

Build basis for layer s: $(F^{s}, W^{s}) := (\widetilde{F}^{s})$

// Columns of $[F \; F^1]$ are linearly independent

$F := [F \; F^s]$

End

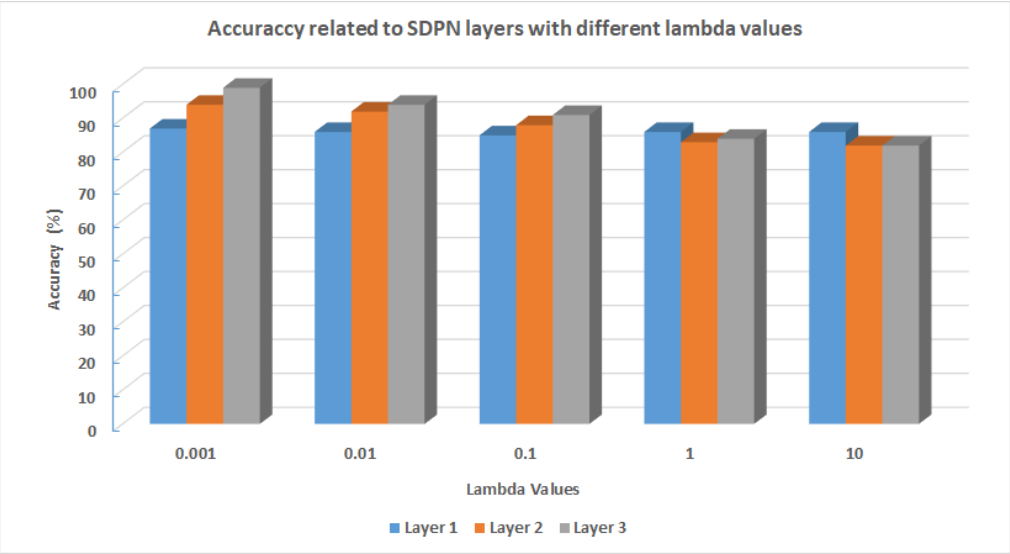

FIGURE 4 Accuracy relating to SDPN layers

To verify that the model doesn't only performed exceptionally well on training data, and it is able to do so on the testing data (avoid over-fitting), we adopt to use L2 regularization technique, In which the loss function is extended by adding the regularization term that contain regularization parameter called lambda value (λ) which controls how to penalize the features. It is a hyperparameter with no fixed value. As its value increases as there will be high penalization for the features. As a result, the model becomes simpler. When its values decrease there will be a low penalization of the features and thus the model complexity increases. Fig. 4 shows the accuracy-related to the SDPN layers using different values of lambda (λ). from the figure we can see

that each layer archive's different accuracy. By experiments we found that the higher accuracy is achieved on the layer 3 with the lambda value equals to (0.001). However, note that in SDPN algorithm we do not need to specify the number of iterations. Instead, we can simply create the layers one-by-one, and then we can check the performance of the resulting network on a validation set, and stop once we reach a satisfactory performance.

With this methodology, all optimal features selected by SMO algorithms are learned in each layer of SDPN, and high-level and optimal features are classified. Building on these significant features, SDPN classifies the data into normal and anomalous, based on the kernel function. Our proposed DL-IDS system performs preprocessing, feature selection and classification for intrusion detection, and the elimination of uncertainties in the dataset improves attack detection performance. Optimal feature selection improves the accuracy of the classifier, and the involvement of SDPN maps optimal low-level features to high-level features.

Fig. 5 shows the comparison of time cost in seconds to build SDPN layers with different values of lambda (λ) before and after using SMO. from this figure we can note the saved time achieved by using the SMO features selection.The SMO reduced the training time on average of 66.59% because the dimension of the dataset is reduced to a low level which boost the classification speed, and the decreased calculations which can decrease the algorithm running time which give us the indication that this model can be used with the real time environments.

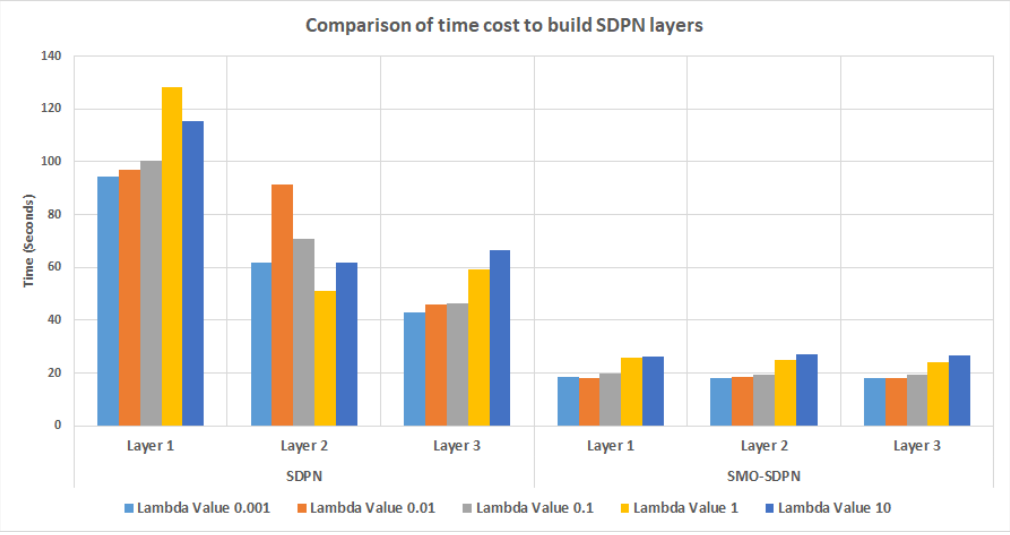

FIGURE 5 Comparison of time cost to build SDPN layers

5 | PERFORMANCE EVALUATION

In this section, we analyze the performance of the proposed DL-IDS system, based on significant performance metrics, such as accuracy, precision, recall and F-score.

5.1 | Dataset Description

To evaluate the performance of DL-IDS we used the NSL-KDD benchmark dataset, which is the most recent version of the KDD'99 dataset[27]. This dataset contains DoS, R2L, probe and U2R Cyber-attacks, and consists of 125,973 records for training and 22,554 records for testing. It involves 41 features, including duration, protocol, service, flag, source bytes and destination bytes.

A brief analysis of the NSL-KDD dataset is provided in Table 3. In DL-IDS, the dataset first undergoes the preprocessing phase in which redundant data are eliminated and missing values are replaced. Then optimal features are selected by the SMO algorithm, and further feature learning and classification with optimal features is performed by SDPN. We used Python-Tensorflow on CPU

TABLE 3 Dataset analysis

Traffic		*Training*	*Testing*
Normal		67343	9711
Attacks	**DoS**	45927	7458
	Probe	11656	2754
	R2L	995	2421
	U2R	52	200
Total		**125973**	**22544**

core i7 based machine to simulate our proposed framework, and for optimal feature selection 41 populations were initialized in the SMO algorithm, and 100 iterations were performed.

5.2 | Performance metrics

In this work, we considered for comparison performance metrics such as accuracy, precision, recall and F score, which are defined as follows:

Accuracy: This is defined as the percentage of correct predictions; that is, the percentage of anomalous traffic that is classified correctly. It is the ratio of correct detections to the total number of records in the dataset, and can be computed as follows,

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{13}$$

Accuracy is computed based on false positive (FP) and true positive (TP) values.

Precision: This metric describes the classifier's ability to predict normal data without conditions. It is defined as the number of true positives divided by the number of true positives, plus the number of false positives as follows:

$$Precision = TP/((TP + FP)) \tag{14}$$

Recall: Recall is the ratio of the number of records correctly classified to the number of all corrected events, and can be computed as follows:

$$Recall = TP/((TP + FN)) \tag{15}$$

Here, FN represents the false negative.

F1-Score: This is defined as the harmonic mean of recall and precision, which can be computed as follows:

$$F1\text{-}Score = 2TP/((2TP + FP + FN)) \tag{16}$$

All these metrics are significant for evaluation of the classifiers.

5.3 | Comparative analysis

In Fig. 6, we show comparative analysis with a deep learning-based attack detection mechanism (D-DL)**?** .In D-DL a deep learning-based Cyber-attack detection mechanism for IoT was distributed over fog nodes using the NSL-KDD dataset. The authors concluded that, though the DL approach outperforms shallow learning methods, the detection efficiency is limited as it requires further analysis on payload-based detection. Based on the average values of each performance metrics of the deep model with the 5-classes (Normal,DoS, Probe, R2L, and U2R) it can be seen that DL-IDS achieves better accuracy, precision recall and F1-Score performance.

Fig. 7 shows comparisons with different Deep Feature Embedding Learning (DFEL) classifiers: Gradient Boosing Tree (GBT), K-Nearest Neighbours (KNN), Decision Tree (DT) and Support Vector Machine (SVM). DFEL architecture was proposed for Intrusion Detection in IoT[29]. it was intended to reduce the data dimensions by using edge-of-deep and transfer learning, and though it minimizes training time, it cannot improve detection accuracy. As DFEL is analyzed with different classifiers, it does not provide better accuracy than any of these classifiers. Involving effectual preprocessing, optimal feature selection and classification improves the accuracy of DL-IDS.

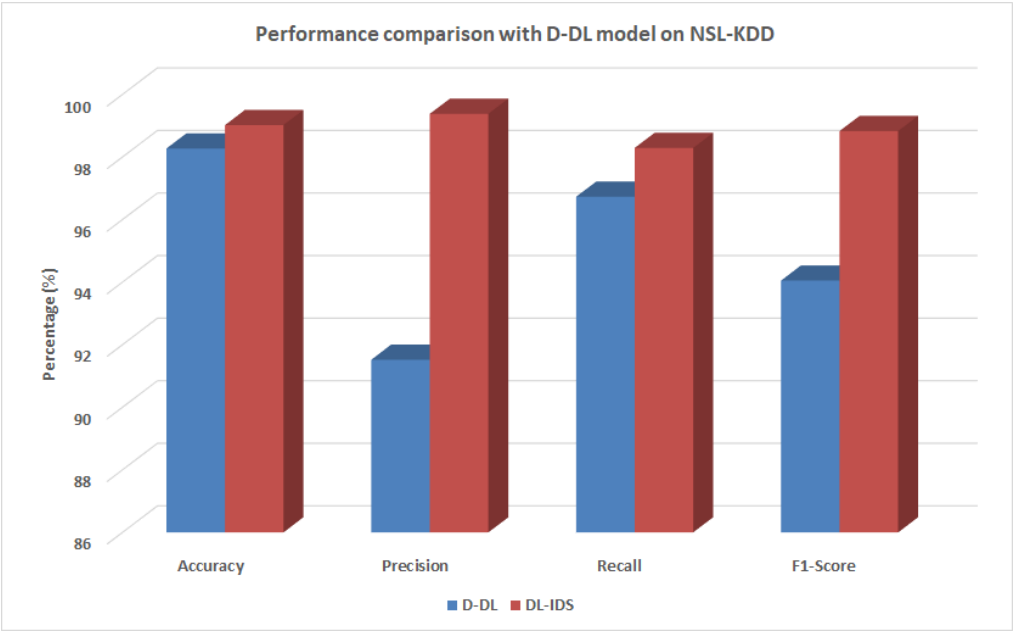

FIGURE 6 Comparison of our model with D-DL

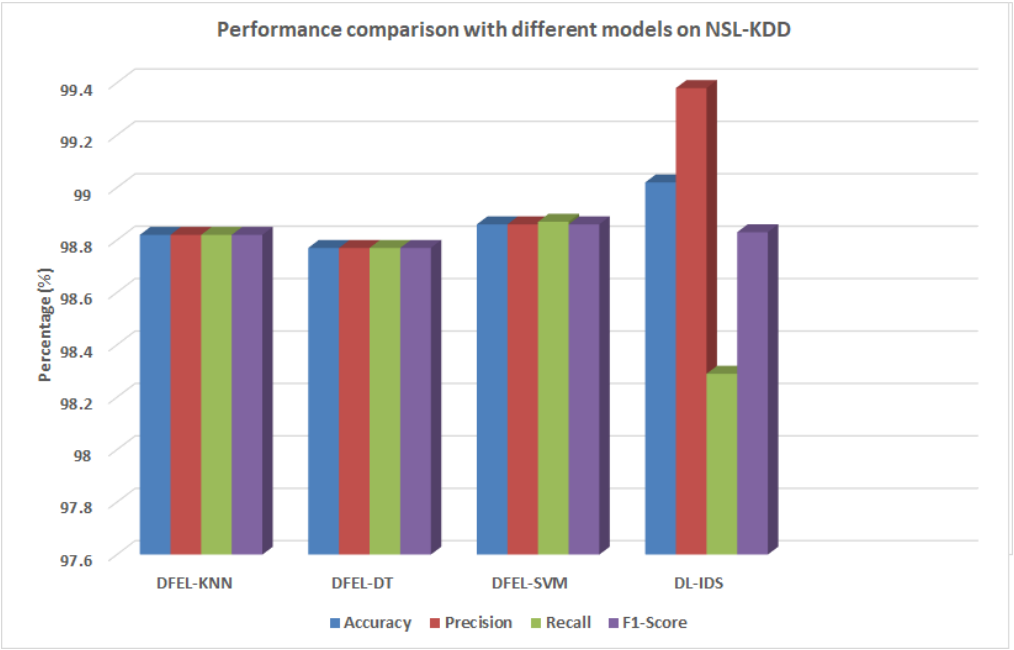

FIGURE 7 Comparison of our model with DFEL

In Table 4, we compare the performance metrics values of our proposed DL-IDS with existing works (DFEL) and distributed DL (D-DL).

Accuracy is an important measure that evaluates the ability of classifiers in intrusion detection. DL-IDS clearly achieves better accuracy than previous approaches, such as DFEL and D-DL. The precision of our proposed work is compared to previous work, and also confirms that DL-IDS outperforms previous work. This is because previous work was unable to predict the class

TABLE 4 Performance comparison on NSL-KDD dataset

Model	*Accuracy (%)*	*Precision (%)*	*Recall (%)*	*F1-Score (%)*
DL-IDS	**99.02**	**99.38**	**98.29**	**98.83**
D-DL	98.27	88.85	96.50	92.52
DFEL GBT	98.54	98.54	98.53	98.53
DFEL KNN	98.82	98.82	98.82	98.82
DFEL DT	98.77	98.77	98.77	98.77
DFEL SVM	98.86	98.86	98.87	98.86

of data since it was processed with non-optimal features and uncertain datasets. We were able to overcome these issues, and thereby achieved high precision.

The recall capability of proposed and previous works is the other performance measure we compared. We found that DL-IDS is also effective in recall metrics, which is significant in intrusion detection. Thus, the proposed DL-IDS can detect all anomalies with no significant errors.

For any classifier, the F1 score is critical to measuring its performance. We compared the DL-IDS F1 score with other models and achieved better F1 scores. Therefore, our proposed DL-IDS with nearest neighbour-based preprocessing, SMO-based feature selection and SDPN-based classification provides better security for IoT environments than previous models.

6 | CONCLUSION

This paper proposes a novel deep learning based intrusion detection system (IDS) for ever-growing IoT based networks to detect severe anomalies. With DL-IDS, A spider monkey optimization (SMO) algorithm is used to extract most relevant features from the dataset. and the stacked deep polynomial network (SDPN) is then used to learn the optimal features and to classify the data as normal or anomalous in different attack categories, e.g., DoS, U2R, R2L and probe attack. We evaluated our DL-IDS system using the NSL-KDD dataset, and our proposed work achieves better performance in accuracy (99.02%), precision (99.38%), recall (98.29%) and F1-score (98.83%). For future research, we plan to evaluate DL-IDS with different classifiers, such as Na ve Bayes, decision tree and random forest, using various datasets, including KDD-99 and UNSW-NB 15.

References

1. J. Howell. Number of Connected IoT Devices Will Surge to 125 Billion by 2030, IHS Markit Says. https://technology.ihs.com, Accessed June 2019.

2. V. Balasubramanian, et al. . A Mobility Management Architecture for Seamless Delivery of 5G-IoT Services. ICC 2019 - IEEE International Conference on Communications (ICC). 2019; 1-7.

3. Al Ridhawi, Ismaeel, et al. . A Profitable and Energy-Efficient Cooperative Fog Solution for IoT Services. IEEE Transactions on Industrial Informatics. (2019).

4. N. Abbas, et al. .A Mechanism for Securing IoT-enabled Applications at the Fog Layer. J. Sens. Actuator Netw. 2019; 8(1):16.

5. SonicWall Inc. . 2019 SonicWall Cyber Threat Report (2019). https://www.sonicwall.com, Accessed June 2019.

6. D.E. Kouicem, A. Bouabdallah, H. Lakhlef. Internet of things security: A top-down survey. Computer Networks.2018;141:199-221.

7. N. Tariq, et al. The Security of Big Data in Fog-Enabled IoT Applications Including Blockchain: A Survey.2019;19(8): 1788.

8. J. Canedo, A. Skjellum. Using machine learning to secure IoT systems, in Proc. 14th IEEE Annu. Conf. Privacy Security Trust (PST). 2016:219-222.

9. M. AL-Hawawreh, N. Moustafa, E. Sitnikova. Identification of malicious activities in industrial internet of things based on deep learning models. Journal of Information Security and Applications. 2018;41:1-11.

10. M. Asad, et al. . DeepDetect: Detection of Distributed Denial of Service Attacks Using Deep Learning. The Computer Journal. 2019;0(0).

11. M. Aloqaily, S. Otoum, I. AlRidhawi, Y. Jararweh. An intrusion detection system for connected vehicles in smart cities. Ad Hoc Networks. 2019.

12. J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao. A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications. IEEE Internet of Things J.2017;4(5):1125-1142.

13. V. Adat, B. Gupta. Security in Internet of Things: issues, challenges, taxonomy, and architecture. Telecommunication Systems.2018;67(3):423-441.

14. R. Doshi, N. Apthorpe, N. Feamster. Machine learning ddos detection for consumer internet of things devices. In Proc. IEEE Security and Privacy Workshops (SPW). 2018.

15. H. Li, K. Ota, M. Dong. Learning IoT in edge: Deep learning for the internet of things with edge computing. IEEE Networks.2018;32(1):96-101.

16. N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, F. Kawsar. An early resource characterization of deep learning on wearables, smartphones and Internet-of-Things devices. in Proc. Int. Workshop on Internet Things Towards Appl.2015:7-12.

17. R. Das, A. Gadre, S. Zhang, S. Kumar, J. Moura. A Deep Learning Approach to IoT Authentication. in Proc. IEEE ICC.2018.

18. H. HaddadPajouh, A. Dehghantanha, R. Khayami, K. Choo. A deep recurrent neural network based approach for Internet of Things malware threat hunting. Future Generation Computer Systems. 2018;85:88-96.

19. B.A. Tama, K.H. Rhee. Attack Classification Analysis of IoT Network via Deep Learning Approach. Research Briefs on Information and Communication Technology Evolution (ReBICTE). Vol. 3, Nov. 2017.

20. P. M. Shakeel, S. Baskar, V. R. S. Dhulipala, S. Mishra, M. M. Jaber. Maintaining security and privacy in health care system using learning based deep-Q-networks. J. Med. Syst.2018;42:186.

21. A. Azmoodeh, A. Dehghantanha, K. Choo, K. Robust. Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning. IEEE Transactions on Sustainable Computing.2019;4:88-95.

22. C.D. Mcdermott, F. Majdani, A.V. Petrovski. Botnet detection in the internet of things using deep learning approaches. in Proc. Int. Joint Conf. on Neural Networks. Aug. 2018.

23. A. Abeshu, N. Chilamkurti. Deep Learning: The Frontier for Distributed Attack Detection in Fog-to-Things Computing. IEEE Communications Magazine.2018;56(2):169-175.

24. J. Bansal, H. Sharma, S. Jadon, M. Clerc. Spider Monkey Optimization algorithm for numerical optimization. Memetic Computing.2014;6(1):31-47.

25. J. Shi, S. Zhou, X. Liua, Q. Zhang , M. Lu, T. Wang. Stacked deep polynomial network based representation learning for tumor classification with small ultrasound image dataset. Neurocomputing .2016;194:87-94.

26. X. Zheng, J. Shi, Y. Li, X. Liu, Q. Zhang. Multi-modality stacked deep polynomial network based feature learning for alzheimer's disease diagnosis. in Proc. IEEE 13th Int. Symp. on Biomedical Imaging (ISBI), , Apr. 2016:851-854.

27. M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. in Proc. IEEE Int. Conf. Comput. Intell. Secur. Defense Appl.2009:1-6.

28. A. Diro, N. Chilamkurti. Distributed attack detection scheme using deep learning approach for Internet of Things. Future Generation Computer Systems.2018;82:761-768.

29. Y. Zhou, M. Han, L. Liu, J. He, Y. Wang. Deep learning approach for cyberattack detection. in Proc. IEEE INFOCOM Workshops. 2018:262-267.

How to cite this article: Yazan O., Dandan L.,and Amiya N. (2019), DL-IDS: A Deep Learning-based Intrusion Detection Framework for Securing IoT, , .