International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2s
DOI: https://doi.org/10.17762/ijritcc.v11i2s.6035
***Article Received**: 22 November 2022 **Revised**: 20 December 2022 **Accepted**: 02 January 2023*

Network Intrusion Detection Method Using Stacked BILSTM Elastic Regression Classifier with Aquila Optimizer Algorithm for Internet of Things (IoT)

Rekha Gangula[1], Dr. Murali Mohan Vutukuru[2] and Dr. Ranjeeth Kumar. M[3]
[1]Research Scholar, Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India-522503.
gangularekha@gmail.com
[2]Associate Professor, Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India-522503.
muralimohan.klu@gmail.com
[3]Assistant Professor, Department of CSE, Kakatiya Institute of Technology and Science, Warangal, Telangana, India-506015

Abstract
Globally, over the past ten years, computer networks and Internet of Things (IoT) networks have grown significantly due to the increasing amount of data that has been collected, ranging from zettabytes to petabytes. As a result, as the network has expanded, security problems have also emerged. The large data sets involved in these types of attacks can make detection difficult. The developing networks are being used for a multitude of sophisticated purposes, such as smart homes, cities, grids, gadgets, and objects, as well as e-commerce, e-banking, and e-government. As a result of the development of numerous intrusion detection systems (IDS), computer networks are now protected from security and privacy threats. Data confidentiality, integrity, and availability will suffer if IDS prevention efforts fail. Complex attacks can't be handled by traditional methods. There has been a growing interest in advanced deep learning techniques for detecting intrusions and identifying abnormal behavior in networks. This research aims to propose a novel network namely stacked BiLSTM elastic regression classifier (Stack_BiLSTM-ERC) with Aquila optimizer algorithm for feature selection. This optimization method computes use of a cutting-edge transition function that enables it to be transformed into a binary form of the Aquila optimizer. A better solution could be secured once number of possible solutions are found from diverse regions of the search space utilizing the Aquila optimizer method. NSL-KDD and UNSW-NB15 are two datasets that enable learning characteristics from the raw data in order to detect harmful prerequisites characteristics and effective framework patterns. The proposed Stack_BiLSTM-ERC achieves 98.l3% of accuracy, 95.1% of precision, 94.3% of recall and 95.4 of F1-score for NSL-KDD dataset. Moreover, 98.6% of accuracy, 97.2% of precision, 98.5 of recall and 97.5% of F1-score.

Keywords: -Intrusion detection, optimization, neural network, Internet of Things (IoT), Regression model.

I. Introduction

Massive improvements in telecommunications networks and the development of the idea on Internet of Things have resulted from incredible changes throughout the ordinary use of electronic services and applications (IoT). The Internet of Things (IoT) is a new revolution for telecommunications wherein equipment acts as entities or "properties" which can communicate to one another, perceive their surroundings, and share data over the Internet [1, 2]. One trillion Ip or other things will be connected to the Internet by 2022 owing to IoT networks [3]. Furthermore, the IoT concept has been employed to develop smart contexts with a wide range of application domains and related services, such as smart cities and smart homes. By addressing issues with the housing conditions, energy usage, and essential goods, such smart settings aim to increase human productivity and comfort [4]. The significant expansion of IoT-based applications and services across various networks is directly related to this goal. A notable benefit of a smart city built on an IoT system is the Padova Smart City in Italy [5]. Devices in intelligent surroundings cooperate to carry out tasks. IPv6, wireless communication methods, and wireless sensors all contribute to the growth of smart surroundings. These environments come in a variety of forms, including smart homes, smart cities, and smart healthcare. Smart surroundings and IoT systems working together increase the efficiency of smart objects [6]. However, IoT systems are vulnerable to a number of security threats, including distributed denial-of-service (DDoS) and denial-of-service (DoS) assaults. Such threats have the potential to seriously harm an IoT network's applications for smart environments and IoT services.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2s
DOI: https://doi.org/10.17762/ijritcc.v11i2s.6035
***Article Received**: 22 November 2022 **Revised**: 20 December 2022 **Accepted**: 02 January 2023*

As a result, IoT system security had grown to be a top priority [7].

IoT networks should establish a new line of defence to defend IoT systems from cyber threats. Unfortunately, due to the above-mentioned special properties of IoT systems, particularly its low energy, pervasive nature, heterogeneity, constrained throughput, and communications technology, standard IDS approaches have been less successful or unsuitable for their protection [8]. The identification of malicious activities, such as those targeting IoT networks, has lately been successfully applied using machine learning (ML) and deep learning (DL) based approaches. This is due to the ability of ML/DL-based algorithms to detect both normal and unusual activity in Distributed systems. To discover typical patterns, it is possible to capture and examine network traffic and IoT devices. Any departure from these established norms can be used to identify abnormal conduct. Additionally, methodologies are based on ML/DL have indeed been developed that anticipate new zero-day attacks. In order to design the security of IoT devices and networks, ML/DL optimization systems provide strong security rules. The following are the accomplishments of such a work:

The preprocessed data is then used in conjunction with the metaheuristic optimization method known as the Aquila optimizer algorithm for feature selection to enhance the recognition rate.

To develop a neural network for anticipating various computing traffic attack types in real-time using a Virtual Machine (VM)-based traffic analysis process.

To create a cutting-edge method for intrusion detection utilizing deep neural networks that uses stacked BiLSTM ElasticNet Regression Model networks to effectively acquire characteristics from the raw data to recognize harmful program sequences and prerequisites patterns. It significantly lowers the price of building artificial characteristics.

The structure of the current document is as follows: A relevant collection of research for IoT malware detection using neural networks is provided in section 2 of the presentation with table. In section 3 suggested Aquila optimization method for feature selection and classifier is given. In part 4, the performance of the suggested model is shown along with a benchmark method. Section 5 presents the general conclusion for the suggested method.

II. Related Work

On identifying threats in IoT systems, many research investigations in the area of IoT security were conducted. Unfortunately, the majority of the research on IoT security that has been done so far has not primarily focused on the use of optimization-oriented ML/DL methods for IoT security. Consequently, this part offers such.

The creation of PODCNN-LWID, or Planet Optimization with a Deep Convolutional Neural Network for Lightweight Intrusion Detection, in a resource-constrained Internet of Things context, is the main topic of [9]. The primary goal of the propose technique is to recognise and classify intrusions. Two important processes—classification and parameterization incorporated in the given model, which uses a Deep convolutional neural model for the intrusion recognition process at the beginning phase. The adam algorithm is then used by the PODCNN-LWID model as a hyperparameter embedding process in the second stage.

To address the issue of complexity and effectively defend the IoT environment, [10] offer an intrusion prevention hyperparameter control system (ID-HyConSys) that regulates the IDS utilizing proximity policies optimization (PPO). A deep neural network (DNN) feature extractor that collects beneficial properties from a dynamic distributed system, a k-means clustering that groups the collected information, and a PPO operator that automate the IDS via retraining and command ID-HyConSys' intrusion detection module.

A deep-convolutional neural network (DCNN)-based IDS is suggested in [11]. Three completely associated dense tiers and two pooling layers make up such model. The presented method seeks to boost efficiency while using less computing power.

A unique strategy for detecting intrusions into IoT networks is proposed in [12] and is based on a convolutional neural network with adaptive particle swarm optimization (APSO-CNN). More specifically, the structure parameters of a one-dimensional CNN are adaptively optimised using the PSO method with variation in inertia weight. The performance index of PSO is determined using the cross-entropy losses feature value of the testing set that was acquired through CNN's first training.

A unique intrusion detection system to find malicious assaults intended for a smart environment is presented in [13]. The suggested intrusion detection approach employs a random forest approach and a correlation tool to identify the key independent factors and create a new neural network-based assault classifier. A shallow neural network and an enhanced neural-based classifier are given to identify malicious attacks.

[14] proposes a new hybrid optimized long short-term memory (LSTM) technique. The IoT channel's spatially and geographically linked characteristics are extracted using a convolutional neural network, and then various attacks are predicted using an efficient LSTM. To further decrease computing cost and improve accuracy rate, firefly swarm optimization is coupled into LSTM.

Authors established novel feature extraction and selection techniques for the Intrusion detection system in [15] by utilizing the benefits of swarm intelligence (SI) algorithms. They create a feature extraction system based on traditional neural networks. After that, utilizing the algorithm, quila optimizer, they provide an alternate feature selection (FS) strategy.

An IOT - based healthcare network is used in [16] to recognize unusual behaviour using a convolutional neural network. The characteristics which are supplied into the convolutional neural network have a significant impact on how accurate its recognition is. Since it has a substantial effect on the learning procedure, choosing the meaningful and discriminative aspects of network traffic is an important and difficult problem. The butterfly optimisation algorithm, a meta-heuristic optimisation algorithm, was utilized in the suggested method to choose the best attributes for an artificial neural network's learning process.

The Levenberg Marquardt (LM) based back propagation (LM-BP) neural network model is depicted in [17]. An intrusion prevention system is equipped with the LM-BP neural network model and the back propagation intrusion detection process is provided. This algorithm uses numeral characteristic to enhance the weight cut-off of the conventional BP neural network since it has the properties of quick iteration velocity and robust design.

An intrusion detection model based on enhanced genetic algorithms (GA) and deep belief networks is presented within [18]. (DBN). In order for the intrusion prevention system based on the DBN to attain a high detection accuracy with such a reduced thickness, confronting various forms of threats, the optimum number of hidden layers and density of neurons within each layer are created dynamically by numerous rounds of the GA.

An effective Intelligence methodology for intrusion detection systems (IDS) in IoT system is suggested in [19]. They made use of deep learning and metaheuristic (MH) method improvements, which have been proven effective at solving challenging engineering challenges. Convolutional neural networks (CNNs) are a feature extraction technique that researchers suggest adopting to extract pertinent characteristics. Additionally, they create a novel technique for feature selection utilizing the operators of the differential evolution (DE) algorithm, which is a new form of the transient search optimization (TSO) algorithm. Table 1 shows literature review summary

Table 1. Literature review summary

Author/year	Method	Merits	demerits
Alissa et al., (2022)	Planet Optimization with a deep convolutional neural network for lightweight intrusion detection (PODCNN-LWID)	Very little samples are needed for retraining.	time-consuming & expensive
Han et al., (2022)	a deep neural network	It demonstrates resistance to unimportant qualities.	The cost of calculation is greater
Ullah et al., (2022)	deep-convolutional-neural-network (DCNN)	Simple and straightforward approach	For dealing with tags, large labelled information is required.
Kan et al., (2021)	Adaptive Particle Swarm Optimization Convolutional Neural Network (APSO-CNN)	It is resistant to being overfit.	sluggish of forecasting engine
Ramaiah et al., (2021)	shallow neural network	outperforms a single classifier in performance	Users need large things to achieve greater results.
Alqahtani et al., (2022)	novel hybrid optimized long short-term memory (LSTM)	It lowers variability.	Network training is challenging
Fatani et al., (2021)	conventional neural networks (CNN)	It needs far less entries and skips the	Additional stages of processing

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2s
DOI: https://doi.org/10.17762/ijritcc.v11i2s.6035
***Article Received**: 22 November 2022 **Revised**: 20 December 2022 **Accepted**: 02 January 2023*

		feature selection step.	are necessary.
Li et al., (2021)	artificial neural network	It generates a more reliable, effective outcome that is overfitting-resistant.	a heavy reliance on claustral Euclidean distance
Yang et al., (2019)	Levenberg Marquardt (LM) based back propagation (LM-BP) neural network model	may lessen the data's complication	Difficulties with vanishing gradients and explosive gradients
Zhang et al., (2019)	improved genetic algorithm (GA) and deep belief network (DBN)	lower cost	It takes time.
Fatani et al., (2021)	transient search optimization (TSO) convolutional neural networks algorithm	requiring fewer time	costly

From the aforementioned techniques, it can be seen that the multi objective optimization challenge, that involves a defined set of many possible alternatives with the goal of optimizing and generating the optimum response, while finding the intrusion has issues with feature selection. The neural network stacking method they created is a combination of different neural networks whose functions work in concert to reduce error and increase precision. Neural networks are trained to develop a high aptitude for tackling particular issues. Once trained, neural networks produce distinct results that are combined to produce a multiclass and binary class result. The suggested BiLSTM elastic regression classifier (Stack BiLSTM-ERC) approaches provide reliable answers to challenging issues while attempting to overcome the limitations of individual networks.

III. Proposed Methodology

Figure 1 represents the system design of proposed method. The information is initially taken from the repository and pre-processed by first transforming categorical variables to statistical information, followed by the extraction of features from the statistical information using the Aquila optimization method, and finally trained with the stacked BiLSTM elastic regression classifier.

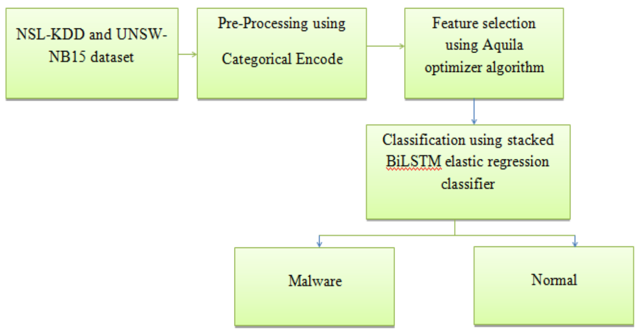

NSL-KDD and UNSW-NB15 dataset
Pre-Processing using Categorical Encode
Feature selection using Aquila optimizer algorithm
Classification using stacked BiLSTM elastic regression classifier
Malware
Normal

Figure 1. Proposed method architecture

3.1 Dataset description

The IXIA Perfect Storm tool in the Digital Range Lab of UNSW Canberra synthesized the unprocessed data packets for the UNSW-NB 15 dataset [20] in order to produce a combination of actual current normal activities and simulated contemporaneous assault behaviours. 200 GB of the raw traffic were captured using the tcpdump utility (e.g., Pcap files). Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms are among the nine attack categories in this dataset. To produce a total of 49 characteristics with class label, the Argus and Bro-IDS tools are employed and 12 methods were built. A division of fully connected information that is 10 attacks-contained and consists of 175343 train connection records and 82337 test connection records. The 42 characteristics in the divided sample have parallel class labels for regular and 9 different attacks. Table 2 details the facts and data for the class of simulated attacks.

Table 2. Description of UNSW-NB15 dataset

Category	Train	Test
Normal	56000	37005
Backdoor	1746	583
Analysis	2000	677
Fuzzers	18185	6062
Shellcode	1133	378
Reconnaissance	10492	3496
Exploits	33393	11132
DoS	12264	4089
Worms	130	44
Generic	40000	18871
Total	175343	82337

The KDDCup99 dataset has been upgraded into the NSL-KDD [21] dataset. In order to produce a much more condensed and effective intrusion detection

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2s
DOI: https://doi.org/10.17762/ijritcc.v11i2s.6035
***Article Received**: 22 November 2022 **Revised**: 20 December 2022 **Accepted**: 02 January 2023*

dataset, scientists from the University of New Brunswick's Canadian Institute for Cybersecurity basically deleted unnecessary information from the original KDDCup99 dataset. The complete NSL-KDD training ("KDDTrain+.TXT") and test ("KDDTest+.TXT") datasets were utilized as our experimental training and test datasets. Because the NSL-KDD dataset is substantially smaller than the complete KDDCup99 dataset, researchers are able to achieve this. Table 3 description of NSL-KDD training dataset.

Table 3. Description of NSL-KDD Training Dataset

	normal	DoS	R2L	U2R	Probe
Frequency	62,378	43,219	897	47	10,946

3.2 Pre-processing of data

The variables containing symbolic properties in the data set are mapped to the digitized feature vector using the categorical coding approach since the model's inputs seems to be digital matrices. Three features of the data set—protocol type, service, and flag—are the major focus of this processing. They each have a Categorical code and have symbol characteristics, correspondingly. Four categorization traits are present: service. Collected are time served, indicators, and procedures. A procedure called "e" transfers each categorical feature's jth value to the jth element of an m-dimensional vector, transforming each feature's m potential categorical values into a value in Rm.

$e(xi) = (0, \dots .1, \dots ,0) if\ xi = j$ $e(xi)$= (0,….1,…,0) if $xi=j$ (1)

Scaling is done for both the categorical and numerical characteristics according to their respective means and standard deviations β.

$n(xi) = \frac{x1-\pi}{\beta}$ n(xi)=(x1-π)/β

(2)

Data transmission is transformed into a sequence of observations during pre-processing, which each observation is expressed as just a feature space. It is possible to mark findings specifying their class, including such "normal" or "anomalous." Consequently, such selected features are appropriate as input for algorithms for machine learning.

3.3 Feature selection using Aquila optimizer Algorithm

Following optimization, feature selection is performed on the validation set through using Aquila optimizer method. The Aquila optimizer (AO) is a recently proposed technology that mimics the natural Aquila hunting process. Following phases make up the hunt procedure: broadened exploration by swooping overhead with a vertically squat, focused exploration by gliding with a contouring fly, broadened exploiting by low-flying descending attack, and focused exploit by strolling and seizing prey. The AO algorithm utilizes a variety of actions that move from the investigation stage to the exploit stage. The very first 2 different repetitions replicate the investigation phase, and the latter third of repetitions imitate the exploiting phase. The AO method is depicted mathematically as follows.

Initialization: The AO method starts by dispersing N options throughout a predetermined range $[Low, Upp][Low, Upp]$ in a D-dimensional lookup tables as indicated in eqn (3).

$Y_{i,j} = Low_j + a \times (Upp_j - Low_j)$

$Y_{i,j} = Low_j + a \times (Upp_j - Low_j)$ (3)

where $Y_{i,j}Y_{i,j}$ is the j-th dimension of the i-th solution, $Low_j Low_j$ and $Upp_j Upp_j$ refer to the lower and upper bound value of the j-th dimension in the search space, and aa is chosen at random from the range of 0 to 1. The position of solutions is kept in matrix $Y_{J\times K}$ $Y_{J\times K}$ Then, by $f(Y_i), f(Y_i)$, the fitness value of each solution is calculated.

Extended observation: An aquila chooses the best hunting area by high-soaring whilst descending vertical once after identifying the victim zone. Due to such activity, the search effort is surveyed at great heights in an effort to determine in which the food might be. Eqn (2) in AO simulates this behaviour, which is carried out when $t_{er} < (2/3 \times Max\_it_{er})$ $t_{er} < (2/3 \times Max\_it_{er})$ and a pseudorandom value $< 0.5 < 0.5$.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2s
DOI: https://doi.org/10.17762/ijritcc.v11i2s.6035
***Article Received**: 22 November 2022 **Revised**: 20 December 2022 **Accepted**: 02 January 2023*

$$Y_1(it_{er}+1) = Y_{best}(it_{er}) \times \left(1 - \frac{it_{er}}{Max_{it_{er}}}\right) + (Y_M(it_{er})$$

$$Y_1(it_{er}+1) = Y_{best}(it_{er}) \times \left(1 - \frac{it_{er}}{Max_{it_{er}}}\right) + (Y_M(it_{er}) - Y_{best}(it_{er}) \times r) \qquad (4)$$

wherein $Y_1(it_{er}+1)$ $Y_1(it_{er}+1)$ is the suitable decision discovered up until the current iteration and roughly resembles the location of the target, and $Y_{best}(it_{er})Y_{best}(it_{er})$ is the answer provided by the prime technique to utilize in the upcoming iteration. The $1 - \frac{it_{er}}{Max_{it_{er}}} 1 - \frac{it_{er}}{Max_{it_{er}}}$ term, is s utilized to regulate the extent of the exploration based on the number of iterations, wherein it_{er} it_{er} denotes the current iteration and $Max_{it_{er}} Max_{it_{er}}$ is the number of iterations that can be performed. In the $it_{er}it_{er}$-th iteration, $Y_M(it_{er})Y_M(it_{er})$ indicates the mean of currently available solutions, as determined by Equation (5),

$$Y_M(it_{er}) = \frac{1}{Num\_sol} \sum_{i=1}^{N} Y_i\ (it_{er}) where\ j = 1,2,3 \ldots L$$

$$Y_M(it_{er}) = \frac{1}{Num\_sol} \sum_{i=1}^{N} Y_i\ (it_{er}) where\ j = 1,2,3 \ldots L \qquad (5)$$

wherein LL indicates the search space's aspect, and Num_solNum_sol the number of possibilities.

Extensive exploration: The second stage is the scavenging technique known as contouring flying of a short glide attack. Aquila hovers over the intended prey, gets ready to dive, and strikes when it is seen from a great height. This performance the Aquila to precisely investigate a given area. Equation (6) in AO simulates this tendency to limit the exploration when the randomly produced value is greater than 0.5 and n $it_{er} < (2/3 \times Max\_it_{er})$ $it_{er} < (2/3 \times Max\_it_{er})$

$$Y_2(it_{er}+1) = Y_{best}(it_{er}) \times levy(G) + Y_R(it_{er}) + (y$$

$$Y_2(it_{er}+1) = Y_{best}(it_{er}) \times levy(G) + Y_R(it_{er}) + (y - x) \times r \qquad (6)$$

wherein, $Y_2(it_{er}+1)Y_2(it_{er}+1)$, $Y_R(it_{er})$ $Y_R(it_{er})$ and $levy(G)\ levy(G)$ stand for the solutions generated by the narrower exploration strategy, a randomly chosen solution from all of the solutions in the $it_{er}it_{er}$-th iteration, and the Levy flight beta coefficient determined by Eqn (7).

$$levy(G) = a \times \frac{c \times C}{e}, C = \frac{\delta(1+\propto) \times sin.\frac{\pi\propto}{2}}{\left(1+\frac{\propto}{2}\right) \times \propto \times 2^{\propto - \frac{1}{2}}}$$

$$levy(G) = a \times \frac{c \times C}{e}, C = \frac{\delta(1+\propto) \times sin.\frac{\pi\propto}{2}}{\left(1+\frac{\propto}{2}\right) \times \propto \times 2^{\propto - \frac{1}{2}}} \qquad (7)$$

wherein cc and ee are random value integers between [0, 1], aa = 0.01, $\propto\propto$ = 1.5,. Formula (8) used to determine y and x, which reflect the helical pattern

$$y = p \times \cos(\theta)\ , x = p \times \sin\ (\theta)$$

$$y = p \times \cos(\theta)\ , x = p \times \sin\ (\theta) \qquad (8)$$

Extended exploit: During phase of extended exploitation, Aquila uses the third strategy to seek food. The Aquila is ready to take flight and launch an attack after meticulously locating the target location. The Aquila descends horizontally and makes the initial stroke to gauge how the prey would react to the attack. $when\ it_{er} > (2/3 \times Max\_it_{er})$ $when\ it_{er} > (2/3 \times Max\_it_{er})$ and a pseudorandom value < 0.5 by Calculation, the action known as a low-flying descent attack is carried out (9).

$$Y_3(it_{er}+1) = (Y_{best}(it_{er}) - Y_M(it_{er})) \times \propto -r + ((Upp - Low) \times r + L) \times \mu$$

$$Y_3(it_{er}+1) = (Y_{best}(it_{er}) - Y_M(it_{er})) \times \propto -r + ((Upp - Low) \times r + L) \times \mu$$

(9)

In which the enlarged exploiting method's solutions are denoted by $Y_3(it_{er}+1)Y_3(it_{er}+1)$ and the exploit modification variables $\propto\propto$ and $\mu\ \mu$ are set to 0.1.

Sharper persecution: The Aquila approach the target and indiscriminately attacks at the smaller exploitation step, which is when the fourth hunting tactic is performed. When $it_{er} > (2/3 \times Max\_it_{er}$ $it_{er} > (2/3 \times Max\_it_{er})$ and a randomly generated value produced by Equation $> 0.5 > 0.5$, the behaviour known as "walking and seizing the prey" occurs (10).

$$Y_4(it_{er}+1) = AF(it_{er}) \times Y_{best}(it_{er}) - (track_1 \times Y(it_{er}) \times r) - track_2 \times levy(G) + r + track_1$$

$$Y_4(it_{er}+1) = AF(it_{er}) \times Y_{best}(it_{er}) - (track_1 \times Y(it_{er}) \times r) - track_2 \times levy(G) + r + track_1 \qquad (10)$$

wherein $Y_4(it_{er}+1)Y_4(it_{er}+1)$ is the existing answer for the iterations, and $Y(it_{er})Y(it_{er})$ is the $(it_{er})(it_{er})$-th searching solutions. In order to balance the search strategy, a quality function known as QF is determined using Equation (11).

$$AF(it_{er}) = t^{\frac{2 \times r - 1}{1 - Max_{it_{er}}}} AF(it_{er}) = t^{\frac{2 \times r - 1}{1 - Max_{it_{er}}}}$$

(11)

The variables $track_1 track_1$ and $track_2 track_2$ depict the motions of the Aquila's prey, with $track_2$

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2s
DOI: https://doi.org/10.17762/ijritcc.v11i2s.6035
***Article Received**: 22 November 2022 **Revised**: 20 December 2022 **Accepted**: 02 January 2023*

$track_2$'s value dropping 2 to 0. Equations (12) and (13) are used to calculate $track_1 track_1$ and $track_2 track_2$

$track_1 = 2 \times r - 1 track_1 = 2 \times r - 1$ (12)

$track_2 = 2 \times (1 - \frac{iter}{Max_{iter}})$

$track_2 = 2 \times (1 - \frac{iter}{Max_{iter}})$ (13)

In order to increase performance of the classifier, minimize computing cost, and boost learning capacity, similar properties in a dataset must be found. Depending on the type of the attribute selection problem, a binary method is used to choose the best subset of attributes. The binary technique uses binary vectors with D entries to represent each solution, where D is the number of features in the dataset. The values of each entry in the solution vector are 0 or 1, with 0 denoting zero decision and 1 denoting the selection of that specific characteristic. Two discrete AO algorithm versions are used to solve the feature selection problem. The feature selection issue is a multi-objective problem that calls for the accomplishment of two opposing goals. There is a contradiction here between reducing the selection of characteristics and subjectively optimizing accuracy. Equation illustrates the weighted sum multi-objective fitness function for assessing each response (14)

$fitness = \sigma\ Class_{error}(F) + \tau \frac{ch\_featu}{tot\_featu}$

$fitness = \sigma\ Class_{error}(F) + \tau \frac{ch\_featu}{tot\_featu}$ (14)

where $\sigma\sigma$ and $\tau\tau$ are two factors that represent the weight of accuracy and the number of selected features and their values are set in the range of σ ∈ [0, 1] and τ = 1 − σ . The classification error, the number of chosen features, and the total number of features are represented by $Class_{error} Class_{error}$, $ch\_featuch\_featu$, and $tot\_featutot\_featu$, respectively.

3.4 Classification using stacked BiLSTM elastic regression classifier

The number of malware activity depending on differential traffic as mentioned in equation (15) was determined by the suggested stacked BiLSTM ElasticNet Regression Model using a logistic regression, as depicted in figure 2:

$P_r(u = V_t) = \frac{1}{1+(V_t,\Omega)} P_r(u = V_t) = \frac{1}{1+(V_t,\Omega)}$

(15)

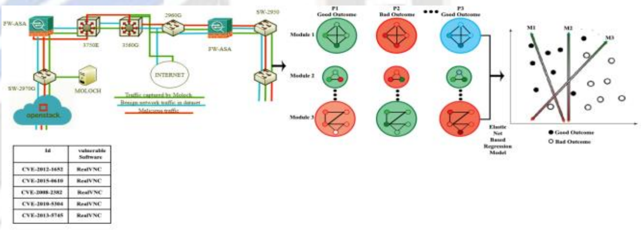
where,

y= {y1, y2, …, yn; yi∈{0, 1}} $u = \{u_1, u_2, \dots u_n\}$ $u = \{u_1, u_2, \dots u_n\}$ and $u_i u_i$ represents the responding variable which includes good or poor outcome labelled as 1 or 0 respectively. The sample server size is represented as n.

$V_t = \{V_{t-1}, V_{t-2}, V_{t-p}\}; V_{t,j} = \{\{V_{t,j(1)}, V_{t,j(2)}, \dots V_{t,j(n)}; j = 1,2, \dots p\}\}.$

$V_t = \{V_{t-1}, V_{t-2}, V_{t-p}\}; V_{t,j} = \{\{V_{t,j(1)}, V_{t,j(2)}, \dots V_{t,j(n)}; j = 1,2, \dots p\}\}.$ Here, V_t V_t indicates the traffic level variables at time t, and p is the total number servers,

$\Omega = \{\Omega_1, \Omega_2, \dots \Omega_p\} \Omega = \{\Omega_1, \Omega_2, \dots \Omega_p\}$ represents regression coefficients in eq. 15 indicating the association degree between response to traffic on every server. Thus, a server having a higher absolute value for $\Omega\ \Omega$ was chosen to be the server with malicious predictivity response.



Id	vulnerable Software
CVE-2012-1652	RealVNC
CVE-2015-0610	RealVNC
CVE-2008-2382	RealVNC
CVE-2010-5304	RealVNC
CVE-2013-5745	RealVNC

Figure 2. Architecture diagram Stacked BiLSTM ElasticNet Regression Model

Furthermore, whenever Ordinary Least Squares (OLS) is utilized, it entails high amounts of data, making it a challenging process to estimate regression coefficients. In order to use OLS, a small number of distinct servers must be selected in advance. Sparse modelling makes the assumption that several regression models were needed to make a prediction, i.e., that some values are non-zero whilst others are zero. Using ElasticNet, regression coefficients are computed as indicated in equation (16) where a penalty term is added to a least-square loss function:

$Arg\ min\ J(y, X(t)) + Arg\ min\ J(y, X(t)) +$ μ $\sum_{j=1}^{p} \frac{wj(1-\propto)1}{2} + \propto . \beta j \sum_{j=1}^{p} \frac{wj(1-\propto)1}{2} + \propto . \beta j$ (16)

Here,

$J(y, X(t)) J(y, X(t))$ indicates loss function of OLS;

$\mu\mu$ is the hyper-parameter for penalty term which is normally set by the user;

$\propto$ is the degree between Ridge $\beta j\,\beta j$ and LASSO $\beta j \beta j$.

$w = \{w_1, w_2, \ldots w_{t,p}\}(w \in> 0)$
$w = \{w_1, w_2, \ldots w_{t,p}\}(w \in> 0)$ are the weights of ElasticNet the selecting bias of every server at time $t\,t$.

Sorting using traffic statistics reveals stable changes over time. ElasticNet uses a server-first, Stability Selection (SS)-selected predictive threat. ElasticNet, which employs SS, then performed a high-selection bias for server placement at time t - 1. As a result, ElasticNet chooses lesser values for the network ratings at t - 1. Iterations of this process were performed over a sustained length of time. As a response, attacks that differed consistently over a wide range of time intervals were found, as shown in formula (17):

$$W(t,j) = W(t,j) = \begin{cases} 1, if\, g(i) \in GL(t-1) \\ \gamma, if\, g(i) \nexists GL(t-1) \end{cases} \begin{cases} 1, if\, g(i) \in GL(t-1) \\ \gamma, if\, g(i) \nexists GL(t-1) \end{cases} \quad (17)$$

Here, $W(t,j)W(t,j)$ symbolizes the weight for the data at the ith server in ElasticNet during time t; $g(i)g(i)$ represents the data at the ith server; $\gamma(\gamma \in> 0;\ \gamma > 1)$ $\gamma(\gamma \in> 0;\ \gamma > 1)$ represents the sampling error, and $GL(t-1)GL(t-1)$ denotes the list of data at time $t-1t-1$. As a result, the data from the server were sorted according to the selection probabilities in descending order, and $SP(final)SP(final)$ is the product derived using selection probability for each server at each t. The probabilities according to each server's frequency of selection were represented by $SP(final)SP(final)$ which was sorted in descending order for all t.

The information collection for the forecasting model were created based on such rankings and utilized in the following stage, as shown in formulas (18) and (19):

$$SP(final) = \{SP1, SP2, SP3.,,,,, SPP\}$$
$$SP(final) = \{SP1, SP2, SP3.,,,,, SPP\} \quad (18)$$

$$SP(j) = \prod_{t=1}^{T} SP(t,j) SP(j) = \prod_{t=1}^{T} SP(t,j) \quad (19)$$

Here, $SP(j)SP(j)$ is the selection probability selection of stability.

Utilizing predictive performance as a variable, which would be calculated at each t using the two situations that are discussed below, we build forecasting model per each data set and assess them at each time step (t).

Case 1: Time-point for model construction = time-point for prediction.

Case 2: Time-point for model construction ≠ time-point for prediction.

In Case 1, the created model is applied to data prediction for comparable time-points. Using the Leave-one-out (LOO) strategy, which uses one data sample as test data and the other for model construction, prediction accuracy (ACC(o)) is evaluated. LOO is applied repeatedly to all samples used as test data.

In instance 2, the model that was built is used to predict data at time points that were not considered when building the model. When building a model, prediction accuracy (ACCT) is calculated using data at t whose mean (ACC (mean)) is determined for each data group at each t as shown in equation (20).

$$(ACC(mean))(ACC(mean)) = \frac{1}{T}\sum_{d\in D} ACC - t(d) + ACC(o)\ \frac{1}{T}\sum_{d\in D} ACC - t(d) + ACC(o) \quad (20)$$

In this case, TT is the duration of each t and $ACCT\ (d) ACCT\ (d)$ is the prediction accuracy for the data at $d.d.$ The time points of $ACC - tACC - t$ are $\boldsymbol{D} = \{t1, tt, \ldots, tT\}\boldsymbol{D} = \{t1, tt, \ldots, tT\}$.

When building the model, the time-point '$t't'$ is not taken into account. Using two different sigmoid ($\sigma\sigma$) functions, the right label and faked label estimations are separated for the model. Only the proper class estimation can be provided back to the server when the accurate label and faked label estimations are separated. In the following expression (21) and (22) the true label term is denoted by the letter "$'c'c$ and the false label by the letter $'f''f'$.

***Article Received**: 22 November 2022 **Revised**: 20 December 2022 **Accepted**: 02 January 2023*

$$L_f = EC_f[logP(C = \frac{C_f}{X_f}(real)]$$

$$L_f = EC_f[logP(C = \frac{C_f}{X_f}(real)]$$

(21)

$$L_{sm} = EY_c(real)[logP(S = \frac{real}{X_C(real)}) + EY_c(fake)[logp(S = \frac{fake}{X_C(fake)}]$$

$$L_{sm} = EY_c(real)[logP(S = \frac{real}{X_C(real)}) + EY_c(fake)[logp(S = \frac{fake}{X_C(fake)}] \quad (22)$$

Thus, the objective function of D is the average of the 3 log-likelihoods that D will maximise, as shown by Calculation (23).

$$L(D) = argmax\ (L_f + L_{sc}).G$$

$$L(D) = argmax\ (L_f + L_{sc}).G \quad (23)$$

Since G G does not produce the proper label samples, the first objective of the D D is to accurately estimate the proper label dispersion from the real samples alone. As not all malware denotes the destructive character and the removal of some right labels is justified in the first layer, the relevance of commands is judged based on their resemblance to the correct label.

```
Algorithm for ElasticNet Regression Model
Output-predicted labels
Compute Pr□{y , Xt }
        y={y1, y2, …, yn; yi∈{0, 1}}: yi
Xt = { Xt,1, Xt,2,…. Xt,p}
β ={β1,β2,….βp}
                y , Xt =server (s)
                obtain LOS of s □k
                        k= Arg min
                Compute the hyperparameter (p) from
LOS
Compute the weight w
w={wt,1, wt,2, …, wt,p}
        for
finalize W(t,i) and GL(t−1)
end for
calculate the selection probability SP(final)
compute mean of prediction accuracies from SP(final)
        σ□ SP(final)
if σ=0 ; correct label otherwise falsified label
```

IV. Performance Analysis

The performance of our proposed stacked BiLSTM elastic regression classifier (Stack_BiLSTM-ERC) is carried out using parameters such as accuracy, precision, recall, F1-score and AUC-score. These parameters are compared with three state-of-art methods such as Planet Optimization with a deep convolutional neural network for lightweight intrusion detection (PODCNN-LWID) [9], hybrid optimized long short-term memory (LSTM) [14], Adaptive Particle Swarm Optimization Convolutional Neural Network (APSO-CNN) [12].

Accuracy is a measure of a deep learning model's total predictive value. The terms true positive (TP) and true negative (TN) describe how well classifier models are able to predict whether an incident will occur or not. False positive (FP) and false negative (FN) indicate how many incorrect predictions the models produced.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (24)$$

Precision assesses the effectiveness of the assault classification model. Precision is the likelihood that a classifier will identify a positive result when a disease is present. It can be calculated as shown in equation (25) and is also referred to as the true positive (TP) rate:

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

$$Precision\ (P) = \frac{TP}{TP+FP} \quad (25)$$

Recall is the likelihood that a classifier will correctly predict a negative outcome in the absence of traffic. It may be calculated as shown in formula (26) and is also known as the true negative (TN) rate.

$$Recall(R) = \frac{TP}{TP+FN}$$

$$Recall(R) = \frac{TP}{TP+FN} \quad (26)$$

To assess the performance of the forecast, the F1-Score is used. It is understood to be the precision and recall's cumulative sum (or harmonized average). The best score is one, while the worst is zero. The TNs are not considered in F-measures. It is possible to calculate the F1-Score as shown in equation (27):

$$F1 - Score = \frac{2*P*R}{P+R} F1 - Score = \frac{2*P*R}{P+R}$$

(27)

4.1 Quantitative performance on NSL-KDD dataset

The table 4 presents the comparison of accuracy for NSL-KDD dataset between existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC

Table 4. Analysis of accuracy for NSL-KDD dataset

classes	PODCNN-LWID	LSTM	APSO-CNN	Stack_BiLSTM-ERC
DoS	89.7	85.6	91.3	98.6
Probe	87.6	88	91.6	97
R2L	87	87.5	92	97.6
U2R	88.4	88.3	92.3	96

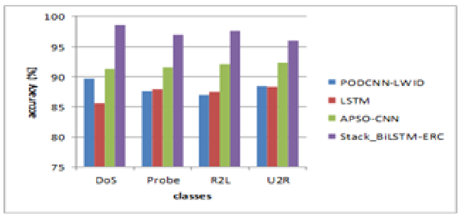


Figure 3. Comparison of accuracy for NSL-KDD dataset

Figure 3 depicts the accuracy for NSL-KDD dataset comparison of existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC. X axis and Y axis shows that various classes and the values obtained in percentage respectively. When compared, existing PODCNN-LWID, LSTM, and APSO-CNN methods achieve 88%,87% and 92.6% of accuracy respectively while the proposed Stack_BiLSTM-ERC method achieves 98.13% of accuracy which is 10.13% better than PODCNN-LWID,9.13% better than LSTM and 5.53% better than APSO-CNN method.
The table 5 presents the comparison of precision for NSL-KDD dataset between existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC

Table 5. Analysis of precision for NSL-KDD dataset

Classes	PODCNN-LWID	LSTM	APSO-CNN	Stack_BiLSTM-ERC
DoS	81.3	87.4	93	97.4
Probe	88	87	93.5	93
R2L	83.4	87.2	92	95.4
U2R	84	87.4	92.1	94

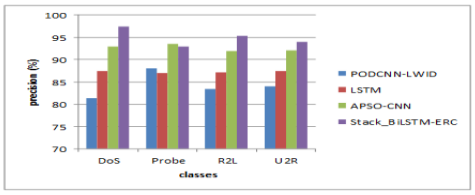


Figure 4. Comparison of precision for NSL-KDD dataset

Figure 4 depicts the precision for NSL-KDD dataset comparison of existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC. X axis and Y axis shows that various classes and the values obtained in percentage respectively. When compared, existing PODCNN-LWID, LSTM, and APSO-CNN methods achieve 84.2%,87.3% and 93.4% of precision respectively while the proposed Stack_BiLSTM-ERC method achieves 95.1% of precision which is 11.1% better than PODCNN-LWID,8.2% better than LSTM and 2.3% better than APSO-CNN method
The table 6 presents the comparison of recall for NSL-KDD dataset between existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC

Table 6. Analysis of recall for NSL-KDD dataset

classes	PODCNN-LWID	LSTM	APSO-CNN	Stack_BiLSTM-ERC
DoS	76.4	89	89.5	91.4
Probe	74.5	86.5	90.2	91.6
R2L	77	88	91.4	92.4
U2R	76.4	87.1	90	94

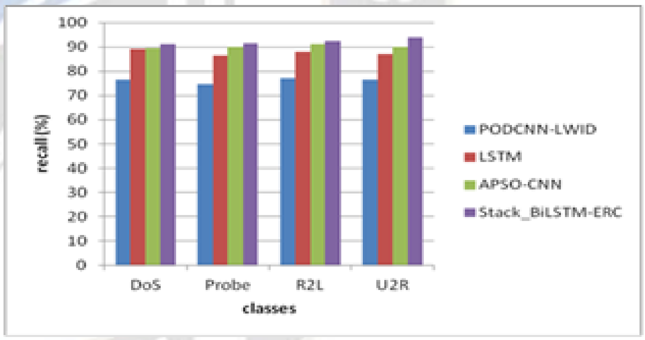


Figure 5. Comparison of recall for NSL-KDD dataset

Figure 5 depicts the recall for NSL-KDD dataset comparison of existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC. X axis and Y axis shows that various classes and the values obtained in percentage respectively. When compared, existing PODCNN-LWID, LSTM, and APSO-CNN methods achieve 76.8%,88.3% and 93.4% of recall respectively while the proposed Stack_BiLSTM-ERC method achieves 94.3% of recall which is 18.5% better than PODCNN-LWID,6% better than LSTM and 3.3% better than APSO-CNN method.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2s
DOI: https://doi.org/10.17762/ijritcc.v11i2s.6035
***Article Received**: 22 November 2022 **Revised**: 20 December 2022 **Accepted**: 02 January 2023*

The table 7 presents the comparison of F1-score for NSL-KDD dataset between existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC

Table 7. Analysis of F1-score for NSL-KDD dataset

classes	PODCNN-LWID	LSTM	APSO-CNN	Stack_BiLSTM-ERC
DoS	81.2	85.5	89.4	93.4
Probe	82	86.4	88.4	95.6
R2L	82.4	86.7	87	92.7
U2R	83	88	87.6	93.4

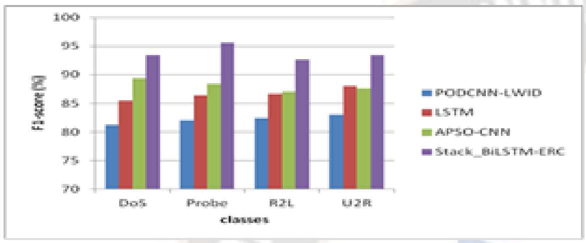


Figure 6. Comparison of F1-score for NSL-KDD dataset

Figure 6 depicts the F1-score for NSL-KDD dataset comparison of existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC. X axis and Y axis shows that various classes and the values obtained in percentage respectively. When compared, existing PODCNN-LWID, LSTM, and APSO-CNN methods achieve 82.3%,86.3% and 87% of F1-score respectively while the proposed Stack_BiLSTM-ERC method achieves 95.4% of F1-score which is 13.1% better than PODCNN-LWID,9.1% better than LSTM and 8.4% better than APSO-CNN method.

4.2 Quantitative performance on UNSW-NB15 dataset

The table 8 presents the comparison of accuracy for UNSW-NB15 dataset between existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC

Table 8. Analysis of accuracy for UNSW-NB15 dataset

classes	PODCNN-LWID	LSTM	APSO-CNN	Stack_BiLSTM-ERC
Backdoor	87.9	81.2	87.4	98.7
Analysis	88	81.4	87.4	98
Fuzzers	87.4	81	88.2	98.5
Shellcode	88.3	81.6	86.4	98.6
Reconnaissance	88.2	81.6	86.3	97
Exploits	88.4	81.7	88.1	97.5
DoS	88.1	82.4	86	97.5
Worms	88.5	82	86.3	98
Generic	88.7	82.5	86.2	98.2

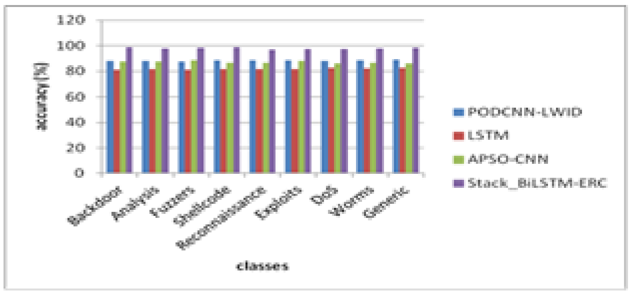


Figure 7. Comparison of Accuracy for UNSW-NB15 dataset

Figure 7 depicts the accuracy for UNSW-NB15 dataset comparison of existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC. X axis and Y axis shows that various classes and the values obtained in percentage respectively. When compared, existing PODCNN-LWID, LSTM, and APSO-CNN methods achieve 82.3%,86.3% and 87% of accuracy respectively while the proposed Stack_BiLSTM-ERC method achieves 95.4% of accuracy which is 13.1% better than PODCNN-LWID,9.1% better than LSTM and 8.4% better than APSO-CNN method.

The table 9 presents the comparison of precision for UNSW-NB15 dataset between existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC.

Table 9. Analysis of precision for UNSW-NB15 dataset

classes	PODCNN-LWID	LSTM	APSO-CNN	Stack_BiLSTM-ERC
Backdoor	83.4	71.2	88.1	97.3
Analysis	83.2	71.6	88	97.5
Fuzzers	82.4	72	88.6	98
Shellcode	83	72.7	87	97.4
Reconnaissance	83.6	72.3	87.5	97.2
Exploits	83.4	73.1	86	97.6
DoS	82	73.7	86.4	98.1
Worms	82.5	73	87	98.3
Generic	82	73.5	87.9	97

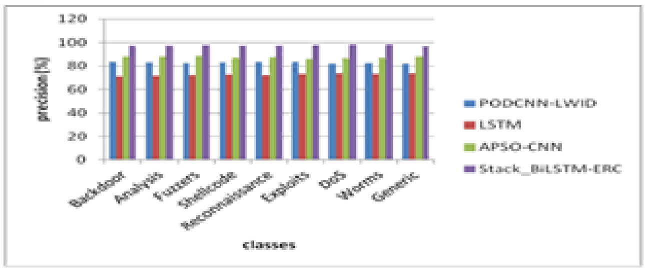


Figure 8. Comparison of precision for UNSW-NB15 dataset

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2s
DOI: https://doi.org/10.17762/ijritcc.v11i2s.6035
***Article Received**: 22 November 2022 **Revised**: 20 December 2022 **Accepted**: 02 January 2023*

Figure 8 depicts the precision for UNSW-NB15 dataset comparison of existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC. X axis and Y axis shows that various classes and the values obtained in percentage respectively. When compared, existing PODCNN-LWID, LSTM, and APSO-CNN methods achieve 83.4%,73.4% and 87.5% of precision respectively while the proposed Stack_BiLSTM-ERC method achieves 97.2% of precision which is 14.2% better than PODCNN-LWID,24.2% better than LSTM and 10.3% better than APSO-CNN method.

The table 10- presents the comparison of recall for UNSW-NB15 dataset between existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC

Table 10. Aanalysis of recall for UNSW-NB15 dataset

classes	PODCNN-LWID	LSTM	APSO-CNN	Stack_BiLSTM-ERC
Backdoor	87	84.3	76.8	98
Analysis	87.4	84.6	78.3	98.6
Fuzzers	87.3	85	78	98.5
Shellcode	86.4	85.9	72.4	98.3
Reconnaissance	86.9	85.7	77	98
Exploits	86.4	82.4	76.3	97.3
DoS	86.1	84.2	75.2	97.2
Worms	86	85.9	75	97.1
Generic	86.7	86	75.8	97

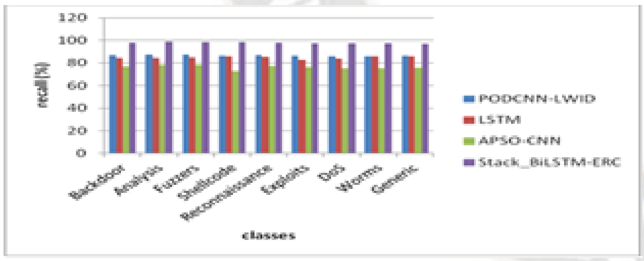


Figure 9. Comparison of recall for UNSW-NB15 dataset

Figure 9 depicts the recall for UNSW-NB15 dataset comparison of existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC. X axis and Y axis shows that various classes and the values obtained in percentage respectively. When compared, existing PODCNN-LWID, LSTM, and APSO-CNN methods achieve 86.4%,85.3% and 78.5% of recall respectively while the proposed Stack_BiLSTM-ERC method achieves 98.5% of recall which is 12.1% better than PODCNN-LWID,12.1% better than LSTM and 20% better than APSO-CNN method.

The table 11- presents the comparison of F1-score for UNSW-NB15 dataset between existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC.

Table 11. Analysis of F1-score for UNSW-NB15 dataset

classes	PODCNN-LWID	LSTM	APSO-CNN	Stack_BiLSTM-ERC
Backdoor	78.3	84.3	76.8	98.5
Analysis	76.4	84.6	78.3	97
Fuzzers	78.4	85	78	97.3
Shellcode	77.8	85.9	72.4	97.2
Reconnaissance	77.3	85.7	77	95.4
Exploits	77	82.4	76.3	96.5
DoS	76.4	84.2	75.2	96.3
Worms	78	85.9	75	98.1
Generic	78.9	86	75.8	94.5

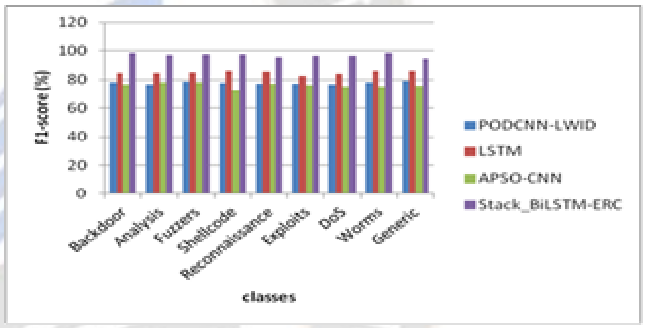


Figure 10. Comparison of F1-score for UNSW-NB15 dataset

Figure 10 depicts the F1-score for UNSW-NB15 dataset comparison of existing PODCNN-LWID, LSTM, and APSO-CNN with the proposed Stack_BiLSTM-ERC. X axis and Y axis shows that various classes and the values obtained in percentage respectively. When compared, existing PODCNN-LWID, LSTM, and APSO-CNN methods achieve 77.5%,86.5% and 75.4% of F1-score respectively while the proposed Stack_BiLSTM-ERC method achieves 97.5% of F1-score which is 20% better than PODCNN-LWID,11% better than LSTM and 22% better than APSO-CNN method.

V. Conclusion

An innovative approach for intrusion detection systems (IDSs) in cloud and Internet of Things (IoT) contexts was presented in this research. The major goal is to develop reliable feature extraction and selection methods by making use of the abundance of deep learning and metaheuristic optimization algorithms. To extract the pertinent characteristics, an aquilla optimization algorithm approach is first recommended. To improve classification accuracy, layered BiLSTM elastic regression classifier is used. Our categorization algorithm performed better even though we were able to analyze samples in less time. Using the NSL-KDD and UNSW-NB15 datasets, which automatically learn attributes from the raw data to capture the harmful file structure patterns and code sequence patterns, we assessed our suggested approach and compared its performance with that of the multiclass and binary class. It is discovered that the suggested classification strategy yields improved outcomes as a result. Future work will focus on Future development will mainly focus on expanding the framework to accommodate new protocols and use cases, such as safeguarding the remote management connection to cloud-based virtual machines.

Acknowledgements
The Department of Computer Science and Engineering at Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India-522503, has supported the authors' work in this area.

References

[1] J. King & AI Awad, (2016) "A distributed security mechanism for resource-constrained IoT devices" , Informatica (Slovenia) Vol. 40, No.1 pp.133–143
[2] M Weber & M Boban (2016) "Security challenges of the internet of things", 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, pp 638–643.
[3] A. Gendreau & M. Moorman, (2016) "Survey of Intrusion Detection Systems towards an End-to-End Secure Internet of Things," 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 84-90.
[4] VP. Kafle, Y Fukushima & H Harai, (2016) "Internet of things standardization in ITU and prospective networking technologies", IEEE Communications Magazine, Vol. 54, No.9, pp.43–49.
[5] A Zanella, N Bui, A Castellani, L Vangelista & M Zorzi (2014) "Internet of things for smart cities", IEEE Internet Things Journal, Vol.1, No.1, pp.22–32.
[6] R Minerva, A Biru & D Rotondi, (2015) "Towards a definition of the internet of things (IoT)-Technical report" , IEEE Internet of Things.
[7] C Han, JM Jornet, E Fadel & IF Akyildiz, (2013) "A cross-layer communication module for the internet of things", Computer Networks, Vol.57, No.3, pp.622–633.
[8] G.Kim, S. Lee & S.Kim, (2014) "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection", Expert System Applications, Vol. 41, pp.1690–1700.
[9] K.Alissa, F.S. Alrayes, K.Tarmissi, A.Yafoz, R.Alsini, O.Alghushairy, & A. Motwakel, (2022) "Planet Optimization with Deep Convolutional Neural Network for Lightweight Intrusion Detection in Resource-Constrained IoT Networks", Applied Sciences, Vol.12, No.17, pp.8676.
[10] H.Han, H.Kim & Y. Kim, (2022) "Correlation between Deep Neural Network Hidden Layer and Intrusion Detection Performance in IoT Intrusion Detection System", Symmetry, Vol.14, No.10, pp.2077.
[11] S.Ullah, J.Ahmad, M. A.Khan, E. H. Alkhammash, M.Hadjouni, Y. Y. Ghadi, & N. Pitropakis, (2022) "A New Intrusion Detection System for the Internet of Things via Deep Convolutional Neural Network and Feature Engineering", Sensors, Vol.22, No.10, pp.3607.
[12] X.Kan, Y.Fan, Z.Fang, L.Cao, N. N. Xiong, D.Yang & X. Li, (2021) "A novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network", Information Sciences, Vol. 568, pp.147-162.
[13] M.Ramaiah, V.Chandrasekaran, V.Ravi & N. Kumar, (2021) "An intrusion detection system using optimized deep neural network architecture", Transactions on Emerging Telecommunications Technologies, Vol.32, No.4, pp.e4221.
[14] S.Alqahtani, (2022) "FSO-LSTM IDS: hybrid optimized and ensembled deep-learning network-based intrusion detection system for smart networks", The Journal of Supercomputing, Vol.78, No.7, pp.9438-9455.
[15] A.Fatani, A.Dahou, M. A. Al-Qaness, S. Lu & M. A.Elaziz, (2021) "Advanced feature extraction and selection approach using deep learning and Aquila optimizer for IoT intrusion detection system", Sensors, Vol.22, No.1, pp.140.
[16] Y.Li, S.M.Ghoreishi & A.Issakhov, (2021) "Improving the Accuracy of Network Intrusion Detection System in Medical IoT Systems through Butterfly Optimization Algorithm", Wireless Personal Communications, pp.1-19.
[17] Yang, Y. Zhuansun, C.Liu, J. Li, & C. Zhang, (2019) "Design of intrusion detection system for internet of things based on improved BP neural network", IEEE Access, Vol.7, pp.106043-106052.
[18] Y.Zhang, P.Li & X.Wang, (2019) "Intrusion detection for IoT based on improved genetic algorithm and deep belief network", IEEE Access, Vol. 7, pp.31711-31722.
[19] A.Fatani, M.Abd Elaziz, A.Dahou, M. A. Al-Qaness & S.Lu, (2021) "IoT intrusion detection system using deep learning and enhanced transient search optimization", IEEE Access, Vol.9, pp.123448-123464.

[20] R. A.Disha & S.Waheed, (2021) "A Comparative study of machine learning models for Network Intrusion Detection System using UNSW-NB 15 dataset", In 2021 International Conference on Electronics, Communications and Information Technology (ICECIT) pp. 1-5.

[21] R. D.Ravipati, & M. Abualkibash, (2019) "Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets-a review paper", International Journal of Computer Science & Information Technology (IJCSIT), Vol.11, No.3.