Received: December 21, 2022. Revised: February 22, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

OptiBiNet_GRU: Robust Network Intrusion Detection System Using Optimum Bi-Directional Gated Recurrent Unit

D. Shankar[1]* **G. Victo Sudha George[2]** **N kanya[1]**

[1]*Department of Information Technology,*
Dr. M.G.R. Educational and Research Institute Chennai, Tamil Nadu 600095, India
[2]*Department of Computer Science and Engineering,*
Dr. M.G.R. Educational and Research Institute Chennai, Tamil Nadu 600095, India
[*] Corresponding author's Email: shankarmtechit@gmail.com

Abstract: Security is considered one of the major challenges in the network due to the increase in network services. Many systems linked to the network play a major role in business and other applications that provide network services. Therefore, it is necessary to identify effective ways to secure the system. One of the significant information security technologies is the intrusion detection system (IDS) that utilizes various deep learning (DL) and machine learning (ML) algorithms to identify network issues. As a result, a novel DL technique is used to classify and detect attacks in the network. To perform this approach, pre-processing is initially done for the collected data using Z-score normalization (ZsN). Then, feature selection is performed using the hybrid pattern search whale optimization algorithm (HPS-WOA). Followed by, network attacks are detected and classified using the optimum bi-directional long short-term memory with the gated recurrent unit (OptiBiNet_GRU). Finally, the hyper parameters such as loss and weight of the proposed DL model are optimized using the pelican optimization algorithm (POA). The effectiveness of the proposed method is validated using two datasets on various performance metrics with some existing baselines. The results show that the proposed method outperforms other methods in terms of accuracy of 99%, precision of 95%, f1-score of 97%, specificity of 99%, and recall of 98% for UNSW-NB15 dataset, whereas accuracy of 99%, precision of 95%, f1-score by 99%, specificity of 99%, and recall of 99% for CICIDS 2017 dataset, respectively.

Keywords: Intrusion detection system, Deep learning, Z-score normalization, Hybrid pattern search whale optimization algorithm, Optimum bi-directional long short-term memory with gated recurrent unit, Pelican optimization algorithm.

1. Introduction

Recently, the emergence of the internet and computer systems have been affected by serious issues such as security, confidentiality, and privacy issues at the time of electronic data transformation [1-3]. In fact, no computer system in the world is fully secure. One effective tool for protecting networks and data is the intrusion detection system (IDS) [4, 5]. The process of monitoring the network or system in order to detect anomalous activities inside the system is known as intrusion detection. IDS can be broadly classified into different types, and the most common types are NIDS (network-IDS) [6] and HIDS (host-IDS) [7]. With the advent of diverse attack types, IDS have been introduced and utilized in numerous network attacks, which permits monitoring a wide range of systems like information systems, networks, and cloud computing systems.

IDS can monitor and identify attacks that compromise the security features, namely confidentiality, integrity, and availability of a system [8]. NIDS aims to recognize the attacks by examining specific events in the network, whereas the HIDS finds the intruders present in individual hosts. The massive technological evolution over the past decades has led to a huge expansion in size and applications being handled by nodes in the network. Moreover, important data is generated in huge amounts and shared among diverse nodes in the network. The security of the network nodes and the data has become the most challenging because of the generation of an enormous number of new attacks [9, 10].

Nearly every node in the network is more vulnerable to security threats. However, the data node is considered the most important for every organization. Any such compromise done to the node's information can cause huge impacts like the decline in market status and financial deficits. In cyber security, NIDS plays a vital role using alerting organizations to overcome malicious activities among devices and networks [11-13]. NIDS can track several network objectives that the HIDS can even miss. Also, HIDS is less capable because it cannot detect particular types of attacks and can't read packet headers. Therefore, NIDS attempts to identify malicious activity such as port scans, DoS (denial-of-service) attacks, and other attacks by observing the network traffic [14, 15]. The existing research on IDS is ineffective in finding various attacks and indicates higher FAR (false alarm rates). This results in the demand for generating accurate, efficient and cost-effective NIDS to offer robust network security.

Nowadays, various methods such as machine learning (ML), Statistical methods, and Deep Learning (DL) can be used to detect intrusions in the form of anomaly detection or misuse detection. ML algorithms allow the IDS to learn and improvise the automatic ability to detect attacks accurately [16, 17]. Some of the commonly used ML algorithms in detecting the intrusion are SVM (support vector machine), RF (random forest), DT (decision tree), NB (naïve bayes) etc. ML algorithms are less effective than deep learning because it often requires more training time and higher false positives, and the dataset's size and dimensionality affect training complexity. Deep learning is the subcategory of ML which offers improved outcomes in intrusion detection [18-20]. The DL approaches used with network NIDS are RNN (recurrent neural network), AE (auto encoder), DBN (deep belief network) etc. DL models involve more hidden layers and are more effective because of the ability to learn the significant features from the dataset on its own to generate the output. This work explores a hybrid DL architecture for intrusion detection with reduced complexity.

1.1 Motivation & problem statement

In the recent era, the vast usage of the Internet leads to the tremendous increase in the huge amount of information exchange among different devices. In modern society, network attacks are one of the most important problems. Security is one of the major concerns when network services are utilized more and more. Traditional approaches like firewall mechanisms focus on data filtering, which may not be suitable for distinguishing the various attack types on time. Many computers linked to the network play vital roles in business, and more applications offer services over the network. Hence, searching for effective techniques to protect the system is necessary. The drawbacks of the existing NIDS are high false alarm rate, response time, low detection rate, unbalanced/imbalanced dataset, high training time, only a few attacks are detected, high FPR (false positive rate), high computational complexity and low convergence. The main motivation of this work is to analyze huge amounts of network traffic data. In this work, a new NIDS concept is introduced as one of the significant information security technologies that use hybrid deep learning techniques to identify network anomalies.

1.2 Key objectives

The main objectives of this research are:

- To develop an intelligent NIDS framework using an optimum bi-directional gated recurrent unit (OptiBiNet_GRU), which automatically detects and classifies the various network cyber-attacks in a timely manner.
- The bio-inspired hybrid feature selection HPS-WOA indicates well-balanced and flexible operator based fine-tuning. It allows the selection of the optimal subset of features from the original network traffic features.
- To evaluate the effectiveness by testing the performance on the challenging datasets, namely UNSW-NB15 and CICIDS 2017 dataset.
- To compare the proposed architecture with the performance of other deep learning classifiers using different metrics.

The rest of the sections in the research work is as follows: Section 2 provides the related work, and section 3 briefly explains the core of the research paper, namely the proposed methodology for detecting and classifying attacks. Section 4 contains the result and discussion, and section 5 conclusion is included for the overall research work.

2. Related work

Some of the previous research on NIDS are discussed as follows:

Vinayakumar *et al.* [21] presented a deep learning model DNN (deep neural network), for the automatic categorization of cyberattacks. With the DNN approach, unpredictable and unforeseen cyberattacks have been detected. The evolution of cyberattacks and the continuously changing network behaviour were the necessary reasons for evaluating the performance with diverse datasets. The network parameters of DNN were optimally chosen using the hyperparameter selection procedures. The learning rate was 0.01-0.5, and the epoch was 1000. The datasets used for experimentation were KDDCup 99, UNSW-NB15, NSL-KDD, WSN-DS, CICIDS 2017 and Kyoto on both binary and multi-class categorization. Also, to execute the deep learning framework in real time, a hybrid model called SHIA (scale hybrid-IDS AlertNet) was developed to monitor the host and network events. The drawback was the high computational cost.

Ravi *et al.* [22] developed a network IDS with deep feature fusion and ensemble meta-classifier for attack detection and categorization of the numerous cyberattacks. The CPS+SDN (cyber physical system with software defined network) was considered the network and environment. The feature extraction was characterized by recurrent deep learning models like LSTM, RNN and GRU. The optimal features were selected utilizing the Kernel PCA (KPCA) approach. Next, the features were fused using the concatenation operation. The detection, as well as classification of network attacks, were processed using the two-stage meta-classifier. The first stage used SVM, RF classifiers were used for prediction, and the second stage used LR for attack classification. The dataset utilized was SDN-IoT. The limitation of the DL model was sensitive to imbalanced data in the network dataset.

Rani and Manisha [23] developed an effectual network IDS concept using DNN multimedia tools and applications to address the class imbalance problem. Some of the problems related to the existing ML based classifiers on NIDS were the time-varying environment, the nature of network data and the complex learning task with the unknown existence of attacks. The data were pre-processed using the min-max normalization procedure. The normalized data were then fed to the DNN, where the modified cross-entropy function was utilized to solve the problem of class imbalance. The challenging datasets utilized for experimentation were UNSW-NB15 and NSL-KDD to indicate the proposed model's superiority. The limitations were overfitting issues and higher training times.

Moizuddin and Victor [24] presented a bio-inspired HDL (hybrid DL) model for identifying network intrusion detection. This IDS model was framed as a two-stage model which includes feature selection with a bio-inspired algorithm and DL based attack classification. Feature selection was processed using the Generalized MGW (mean grey wolf) algorithm, and attacks were categorized based on ElasticNet CAE (contractive auto encoder). The datasets used were NSL-KDD and BoT-IoT, focusing the classification on binary and multi-class models. The overall accuracy gained was 0.999 for multi-class and binary classes. The limitation was computational complexity.

Khan [25] presented a hybrid NIDS framework with deep learning architecture named HCRN (hybrid convolutional recurrent network). This hybrid model allows automatic malicious threat detection at an exact time. The HCRN framework allows intrusion detection that predicts as well as classifies anomalous network-cyberattacks. This hybrid model combines convolutional (CNN) and recurrent (RNN) networks. The CNN allows the convolution function to capture the local features, while the RNN aims to capture the temporal information features. The dataset utilized for the experimentation was CSE-CIC-IDS2018. The accuracy obtained with the HCRN model was 97.75%, and the FAR was 1.4. The drawback was the proposed HCRN has only used a single dataset for testing.

Mendonca *et al.* [26] introduced a fast hierarchical DCNN (deep convolutional neural network) model to detect intrusions. The presented DCNN was based on the hierarchical algorithm Tree-CNN with SRS (soft-root-sign) activation function. This deep learning model detected various attacks such as infiltration, DDoS, web and brute force attacks, which minimizes the training time. The performance was tested with other activation functions like ReLU and Softmax. The dataset used was CICIDS2017. The accuracy obtained with Tree-CNN-SRS was 0.98, and the execution time was 1201 sec. Also, the model was tested on dual actual network intrusion scenarios, such as a university campus and a medium-sized company. The drawback was the suggested model used a single dataset for testing. Table 1 shows a comparative analysis of network intrusion detection using DL models.

3. Proposed methodology

For effective handling and timely identification of various attacks, IDS based on DL algorithms are very effective in efficiently processing large data for identifying any malicious activity. Therefore, a novel DL model is proposed to detect various attacks in the network. Initially, pre-processing is done for the given datasets using a technique known as Z-score

Table 1. Comparative analysis of network intrusion detection using DL models

Author and Year	Method	Merits	Demerits
Vinayakumar *et al.* [2019]	DNN	The network parameters of DNN were optimally chosen using the hyperparameter selection procedures	High computational cost
Ravi *et al.* [2022]	Deep feature fusion and ensemble meta-classifier	The detection, as well as classification of network attacks, was processed using the two-stage meta classifiers.	The model was sensitive to imbalanced data in the network dataset
Rani and Manisha [2022]	DNN	The method uses a cross-entropy function to solve the problem of class-imbalance.	Overfitting issue and higher training time.
Moizuddin and Victor [2022]	HDL	The method focused on classifying both the binary and multi-class models	Computational complexity
Khan [2021]	HCRN	The method automatically detects the malicious threat at the exact time	The method has only used a single dataset for testing
Mendonca *et al.* [2021]	DCNN	The method has experimented on dual actual network intrusion scenarios.	The model has used a single dataset for testing

normalization (ZsN) to convert the traffic data file to values between 0 and 1. Then, a feature selection is made to enhance the system's performance using HPS-WOA, an amalgamation of two optimization algorithms, PS and WOA. Next, the various attacks are classified using hybrid neural networks known as OptiBiNet_GRU, an amalgamation of Bi-LSTM and GRU DL models. Finally, the loss and weight present in the neural network are reduced using an optimization algorithm known as POA. The proposed methodology is shown in Fig. 1.

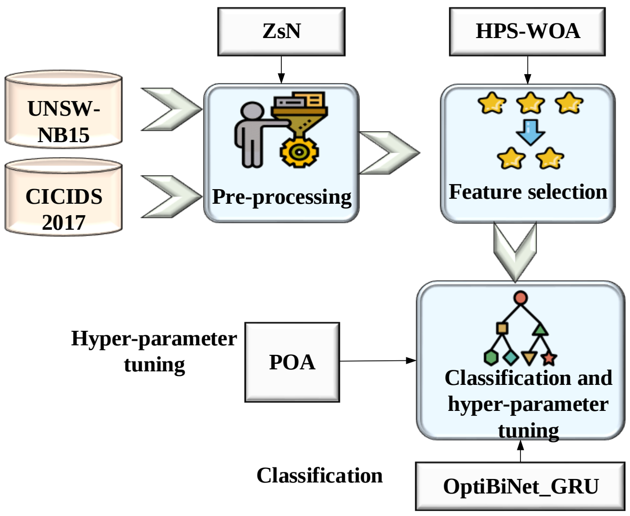


Figure. 1 Proposed methodology

3.1 Pre-processing

In this phase, the pre-processing of the traffic data is performed. The dataset includes various data types, such as the numeric and symbolic representation of data related to services and duration. In this phase, the Z-score normalization (ZsN) technique is applied to the numeric traffic data file to convert the value between 0 and 1. It is utilized to normalize the values of attributes such that after normalization, the mean and the standard deviation become zero and one. Hence, for this characteristic of the normalization z-score is otherwise known as zero mean normalization. Initially, the data should be encoded by utilizing a label encoder to transform any categorical features into numerical ones. After that, the data normalization is performed by computing the normalized value,

$$a_{normalize} = \frac{a_i - \mu}{\sigma} \tag{1}$$

3.2 Feature selection

The major challenge, especially in network systems, where the high dimensionality of network traffic data makes data sparse in hyper-space, which restricts the different algorithms scaling and generalization capabilities. One of the effective ways is to mitigate this issue with the use of feature selection techniques to reduce the dimensionality of data. Feature selection is selecting the optimal subset of features from a large feature set to enhance detection accuracy and performance. Therefore, this work presents an effective network intrusion detection using a wrapper feature selection based hybrid deep learning model, i.e. hybrid pattern search whale optimization algorithm (HPS-WOA) to select the optimal subset of features from the original network traffic features.

3.2.1. Whale optimization algorithm

In this subsection, initially, the inspiration for the whale optimization algorithm is discussed. After that, the mathematical expressions are provided.

3.2.1.1. Inspiration

Whales are considered fancy creatures and are said to be the world's biggest mammals. An adult whale can have the ability to grow up to 30 m long and 180 t weight. The seven various major giant mammal species are killer, Minke, Sei, humpback, right, finback, and blue. It is mainly examined as a predator. From the surface of the oceans, the whales have to breathe, so it is never sleeping. In reality, the whales are only half asleep and highly intelligent animals, even with emotions.

The whales live in groups or alone but are mostly noticed in groups only. Over their lifetime, some species can live in a family. Humpback whales are one of the biggest baleen whales. The favourite prey of whales is small fish herds and krill. The interesting fact about humpback whales is their special hunting strategy. This foraging characteristic is known as the bubble-net feeding approach. The small fishes or schools of krill are preferred humpback whales for hunting which is very close to the surface. It forms distinctive bubbles along with a circle or nine shaped paths.

The unique behaviour called bubble-net feeding is observed only in humpback whales. In order to perform an optimization, the spiral bubble-net feeding maneuver is modelled mathematically.

3.2.1.2. Mathematical model

This subsection provides the mathematical model for prey encircling, spiral bubble web foraging maneuver and prey searching.

Encircling prey: The location of the prey is recognized by the humpback whales and encircles them. In the search space, the optimal structure position is not investigated a priori. The current best candidate solution is the target prey assumed by the WOA algorithm. After defining the best search agent, another search agent will try to update its position towards the best search agent. The mathematical representation of his behaviour is mentioned as follows:

$$\vec{R} = \left|\vec{Q}.\vec{A}^{*}(t) - \vec{A}(t)\right| \tag{2}$$

$$\vec{A}(t+1) = \vec{A}^{*}(t) - \vec{A}.\vec{R} \tag{3}$$

The vectors$\vec{P}$and $\vec{Q}$are computed as below:

$$\vec{\boldsymbol{P}} = \mathbf{2}\vec{\boldsymbol{p}} \cdot \vec{\boldsymbol{s}} - \vec{\boldsymbol{p}} \tag{4}$$

$$\vec{\boldsymbol{Q}} = \mathbf{2} \cdot \vec{\boldsymbol{s}} \tag{5}$$

Where, the iterations $\vec{p}$is minimized linearly from 2 to 0 in both exploration and exploitation stages, and the random vectors range 0 and 1 are denoted as $\vec{s}$.

Update the search agent position based on the position of the current best record (A^{*}, B^{*}). By altering the value of $\vec{P}$and $\vec{Q}$ vectors, to the current position, achieved various places around the best agent. Eq. (3) allows any search agent to update its position in the current best solution neighbourhood and simulate encircling the prey.

The same procedure is developed for a search space with dimensions m and in a hyper-cubes manner, the search agent will move around the obtained best solution. Humpback whales use the bubble-net procedure to attack their prey.

Exploitation stage (Bubble-net attacking strategy): There are two methods are followed to mathematically model the humpback whales' bubble-net behaviour, which is discussed below:

- **Shrinking encircling method:**

It is obtained by minimizing the value of $\vec{p}$in Eq. (4). The fluctuation range of $\vec{P}$is also minimized by$\vec{p}$. In the interval $[-p, p]$, the$\vec{P}$ is a random value here. Over the course of iterations 2 to 0, the value of $p$is minimized. For $\vec{P}$in$[-1,1]$, sets the random values and then defines the new position of a search agent anywhere among the actual position of the agent and the current best agent position.

- **Spiral updating position**

Initially, it computes the distance between the whale located at (A, B) and the prey located at (A^{*}, B^{*}). Among the whale and the prey position, a spiral equation is formed to mimic the helix-shaped motion of humpback whales.

$$\vec{A}(t+1) = \vec{R}'.d^{cr}.cos \rightleftarrows (2\pi r) + A^{*}(t) \tag{6}$$

Where, $\vec{R}' = \left|\vec{A}^*(t) - \vec{A}(t)\right|$denotes the distance of ith whale to the prey.

Within the shrinking circle, the humpback whales swim around the prey and along a spiral-shaped path. Let us consider a 50% probability of selecting the shrinking encircling method or the spiral method to update the whale's position during optimization. The mathematical expression is given as:

$$\vec{A}(t+1) = \begin{cases} \vec{A}^*(t) - \vec{A}.\vec{R} if b < 0.5 \\ \vec{R}'.d^{cr}.cos(2\pi r) + \vec{A}^*(t) if b \geq 0.5 \end{cases} \tag{7}$$

Moreover, humpback whales search randomly for prey. The mathematical structure of the search is mentioned below.

Exploration stage (Search for prey): The same method depends on the difference of $\vec{P}$vector used to search for prey. Based on the position of each other, the humpback whales randomly search. To force the search agent to migrate far away from a reference whale, utilize $\vec{P}$with random values greater than or less than 1 and -1 respectively. In the exploration stage, update the position of the search agent based on the randomly selected search agent instead of the best search agent. The parameter $\left|\vec{P}\right| > 1$highlight the exploration and allows the optimization method to perform a global search.

$$\vec{R} = \left|\vec{Q}.A_{rand} - \vec{A}\right| \tag{8}$$

$$\vec{A}(t+1) = \bar{A}_{rand} - \vec{A}.\vec{R} \tag{9}$$

This optimization method starts with random solutions set. During every iteration, the search agents update their positions according to the randomly selected search agent or the obtained best solution. To provide exploration and exploitation, the parameter $\vec{p}$is minimized from 2 to 0. If the value of $|\vec{p} > 1|$a random search agent is selected, the best solution is chosen. If$\left|\vec{P}\right| < 1$, update the position of the search agents. Based on the b value, this optimization approach can have the ability to switch between a spiral or circular movements. Finally, the termination condition gets satisfied means that the WOA method is terminated.

3.2.2. Pattern search (PS)

PS is a derivative-free evolutionary optimization algorithm that is well suited to solving various optimization problems outside the scope of standard optimization algorithms. The algorithm's advantages are that it is easy to implement, simple in concept, and computationally efficient. To enhance and fine tune

Table 2. Pseudocode for PS algorithm

Start
Step 1: Generate vectors of positions [0 1], [-1 0], and [0 -1]
Step 2: Produce mesh points by incrementing the direction vectors to a_0
Step 3: Generate initial point I_p as $a_0 + [0,1] a_0 + a_0 + [-10]$, and$x_0 + [0 - 1]$
Step 4: At each mesh point
Step 5: Compute objective function
Step 6: if
Step 7: $O_f < I_p$
Step 8: Select it for next iteration and name it as a_1
Step 9: $a_1 \times 2$ (Expansion factor)
Step 10: Repeat steps 4 to 9 until the termination criterion is reached
Step 11: if
Step 12: $O_f <\neq I_p$reached
Step 13: Reuse the I_p
Step 14: $a_1 \times 0.5$ (Contraction factor)
Step 15: Repeat the steps until the termination criterion is reached
end

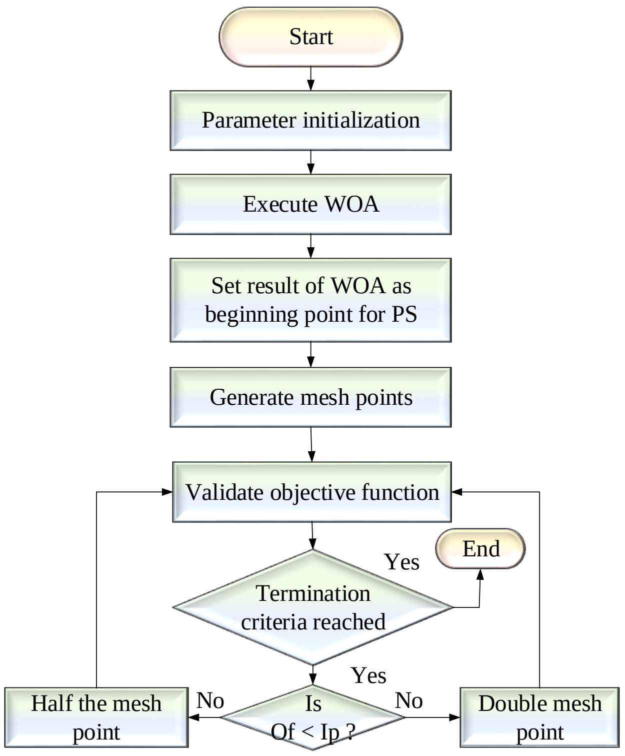


Figure. 2 HHPS-WOA for feature selection

local search and global search, the algorithm consists of a well-balanced and flexible operator. Initially, the optimization algorithm starts with a set of points known as a mesh around the initial points. The WOA issues the current or the initial points. The mesh is generated by incrementing the current point to a scalar multiple of a group of vectors known as a pattern. In the mesh, if a point possesses the best objective function value, it becomes the starting point at the consecutive iteration. The pseudocode for PS is shown in Table 2.

3.2.3. HPS-WOA

The optimization algorithms WOA and PS are combined to form the best optimized feature set. A fitness function can be framed using both optimization algorithms. The fitness function must receive an optimized set of feature vectors and is represented as,

$$F = optimized(f_v) \tag{10}$$

The flowchart for the proposed HPS-WOA for feature selection is shown in Fig. 2.

3.3 Classification using optimum bi-directional long short-term memory with the gated recurrent unit (OptiBiNet_GRU)

3.3.1. Bi-directional long short-term memory (Bi-LSTM)

An exclusive variant of RNN is Bi-LSTM, which consists of two layers of LSTM integrated. One of the LSTM units performs inputs in the forward direction, whereas the other LSTM unit executes inputs in the backward direction. It is also said to expand the traditional LSTM, conserving future and past information. This results in a considerable amount of increment of the information accessible to the network. Thus, a better understanding of the context is achieved by the network. The basic architecture of Bi-LSTM is shown in Fig. 3.

Each LSTM block follows an internal mechanism. The hidden state value is executed and updated based on internal equations and is represented as,

$$\vec{g}_{t_s} = \sigma\left(\vec{p}_g\left\lfloor \vec{h}_{s-1}, \vec{E}_{t_s}\right\rfloor + \vec{a}_g\right) \tag{11}$$

$$\vec{m}_{t_s} = \sigma\left(\vec{p}_m\left\lfloor \vec{h}_{s-1}, \vec{E}_{t_s}\right\rfloor + \vec{a}_m\right) \tag{12}$$

$$\vec{n}_{t_s} = \sigma\left(\vec{p}_n\left\lfloor \vec{h}_{s-1}, \vec{E}_{t_s}\right\rfloor + \vec{a}_n\right) \tag{13}$$

$$\vec{k}_{t_s} = tanh\left(\vec{p}_k\left\lfloor \vec{h}_{s-1}, \vec{E}_{t_s}\right\rfloor + \vec{a}_k\right) \tag{14}$$

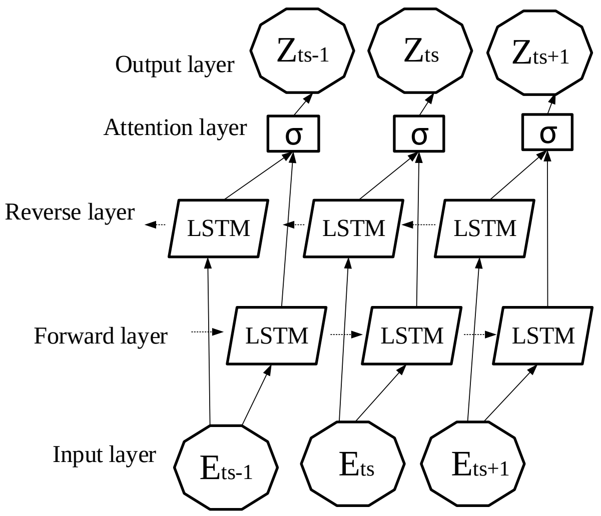


Figure. 3 Bi-LSTM architecture with the forward and backward operation

$$\vec{k}_{t_s} = \vec{g}_{t_s} \otimes \vec{K}_{t_s-1} \oplus \vec{m}t_s \otimes \vec{k}_{t_s} \tag{15}$$

$$\vec{h}_s = \vec{n}t_s \otimes tanh\left(\vec{K}t_s\right) \tag{16}$$

The output hidden layer for the overall forwarder layer LSTM at the time step$t_s$is represented as,

$$\vec{h}_s = g\left(\vec{E}_{t_s}, \vec{h}_{s-1}, \vec{\omega}LSTM\right) \tag{17}$$

Here, the internal operation of the real state forward LSTM unit is denoted as$\vec{\omega}$ $LSTM$. Therefore, inside a backward layer, the internal updates take place and are represented as,

$$\overleftarrow{g}_{t_s} = \sigma\left(\overleftarrow{p}_g\left\lfloor \overleftarrow{h}_{s-1}, \overleftarrow{E}_{t_s}\right\rfloor + \overleftarrow{a}_g\right) \tag{18}$$

$$\overleftarrow{m}_{t_s} = \sigma\left(\overleftarrow{p}_m\left\lfloor \overleftarrow{h}_{s-1}, \overleftarrow{E}_{t_s}\right\rfloor + \overleftarrow{a}_m\right) \tag{19}$$

$$\overleftarrow{n}_{t_s} = \sigma\left(\overleftarrow{p}_n\left\lfloor \overleftarrow{h}_{s-1}, \overleftarrow{E}_{t_s}\right\rfloor + \overleftarrow{a}_n\right) \tag{20}$$

$$\overleftarrow{k}_{t_s} = tanh\left(\overleftarrow{p}_k\left\lfloor \overleftarrow{h}_{s-1}, \overleftarrow{E}_{t_s}\right\rfloor + \overleftarrow{a}_k\right) \tag{21}$$

$$\overleftarrow{k}_{t_s} = \overleftarrow{g}_{t_s} \otimes \overleftarrow{K}_{t_s-1} \oplus \overleftarrow{m}t_s \otimes \overleftarrow{k}_{t_s} \tag{22}$$

$$\overleftarrow{h}_s = \overleftarrow{n}t_s \otimes tanh\left(\overleftarrow{K}t_s\right) \tag{23}$$

The output hidden layer for the overall backward layer LSTM at the time step$t_s$is represented as,

$$\overleftarrow{h}_s = g\left(\overleftarrow{E}_{t_s}, \overleftarrow{h}_{s-1}, \overleftarrow{\omega}LSTM\right) \tag{24}$$

Here, the internal operation of the real state backward LSTM unit is denoted as$\overleftarrow{\omega}LSTM$. The final

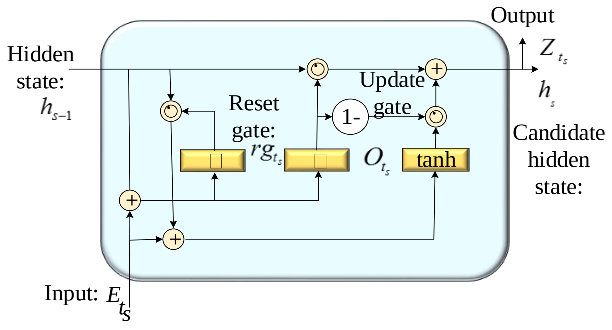


Figure. 4 GRU architecture

output of the hidden layer integrates the hidden states of the forward and backward layers of the Bi-LSTM model along with the activation function. Hence, the past and future data are traced to the output of the bi-LSTM. It possesses more parameters than the GRU and LSTM networks. Due to its capacity to handle complicated data, Bi-LSTM with GRU is widely used in time series detection. The inclusion of Bi-LSTM with GRU enhances the long-term detection accuracy of the proposed model due to its capacity to attain both future and past data.

3.3.2. Gated recurrent unit (GRU)

GRU is one of the advanced versions of RNN (recurrent neural network) and consists of a gating mechanism. It can prevent the vanishing gradient issue that occurs in traditional RNNs. GRU is closer to the traditional LSTM model but only with a few training parameters. GRU is faster than LSTM, with less memory consumption and training time. Without the help of any control, the GRU can expose the entire hidden states. GRU is a more appropriate model of the RNN version for a shorter sequence of input data. The basic architecture of GRU is shown in Fig. 4.

Unlike LSTM, two gates are present in GRU: update and reset gates. The update gate aims to combine the functionality of both the input and forget gates of LSTM. It helps the model determine how much data can be moved into the future based on the previous time steps. A reset gate aims to identify the amount of data that can be erased from the past. Also, GRU detaches the cell's state in LSTM, and the data is transmitted via the hidden state. The components contained in the basic GRU architecture are stated below.

- ✓ An external input dataE_{t_s}at the time step t_s
- ✓ A hidden stateh_{s-1}at the previous time stept_s-1.
- ✓ A reset gate vectomanages the amount of past information that is forgotten after a point-by-point multiplication byh_{s-1}.
- ✓ The output of an update gate with a sigmoid function O_{t_s}identifies the amount of information of the previously hidden layer that will proceed along with the consecutive hidden state.
- ✓ A candidate activation vectorC_{t_s}that utilizes a reset gate vector rg_{t_s} to conserve the past important information.
- ✓ A consecutive hidden state value$h_s$is the same as the outputO_{t_s}.

Each candidate activation vector and internal gate inside a GRU cell for each new input data at a time t_s executes some important updates and are represented as,

$$O_{t_s} = \sigma\left(P_O E_{t_s} + Q_O h_{s-1} + a_O\right) \quad (25)$$

$$rg_{t_s} = \sigma\left(P_{rg} E_{t_s} + Q_{rg} h_{s-1} + a_{rg}\right) \quad (26)$$

$$C_{t_s} = tanh\left(P_C E'_{t_s} + Q_C\left(rg_{t_s} \otimes h_{s-1}\right) + a_C\right) \quad (27)$$

By manipulating the update gate vector, the next hidden layer and the final output of the GRU cell unit can be obtained and is given as,

$$h_s = O_{t_s} \otimes C_{t_s} + \left(1 - O_{t_s}\right) \otimes h_{s-1} \quad (28)$$

The proposed work integrates a GRU layer between the Bi-LSTM layers. After proper tuning, the output neuron layer of the GRU is selected, and it is set to 64 units.

3.3.3. OptiBiNet_GRU

The proposed Bi-LSTM-based GRU model is shown in Fig. 5. It consists of four layers. The first layer is the input layer. The second layer is the Bi-LSTM elements. It is used to decrease the overall training parameters, computational cost and overcome the overfitting problem. The third layer is the GRU. The layer is responsible for learning the temporal relationship between the network flows and adapting the temporal dynamics due to the framework of Bi-LSTM recurrent bidirectional. The layers Bi-LSTM and GRU are the core layers used to classify attacks from the network flow. Then, a flattened layer is introduced to adjust the dimensionality of the input to the following dense layers. Finally, the dense layers are considered the output layers in which the attack category is classified. Since the proposed work uses two datasets such as UNSW-NB15 and CICIDS 2017. Therefore, classification is done with both datasets to detect the

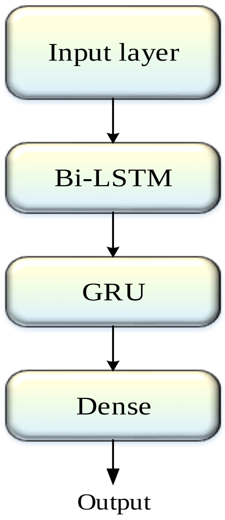


Figure. 5 OptiBiNet_GRU

category of attack. The attack classes are normal, exploits, reconnaissance, DoS, generic, shellcode, fuzzers, worms, backdoors, and analysis for the first dataset. The other 10 classes are benign, DDoS, portscan, bot, infiltration, DoS slowloris, DoS slowhttptest, DoS hulk, DoS goldeneye, and heartbleed. The detection loss for multi-class classification is determined using the categorical cross entropy and is represented as,

$$L(p_d, e_d) = -\sum_y p_d(y)\, log\big(e_d(y)\big) \tag{29}$$

The loss and weight parameters of the proposed model are tuned using an effective pelican optimization algorithm (POA).

3.4 Hyper-parameter tuning using pelican optimization algorithm (POA)

The loss and weight contained in the hybrid neural networks are reduced using a novel optimization algorithm known as POA. It is a swarm-based optimization algorithm used to solve a large amount of loss reduction problems. It is a group of birds with long beaks and a large bag in their throats to catch and swallow food sources. These birds are known as skilled hunters due to their intelligent behaviour when hunting prey. The objective function for the proposed algorithm is shown below.

Objective function: The main objective function is to reduce the loss and weight of the neural networks used. A lot of weight loss happens as the proposed work uses hybrid neural networks. If the loss and weight of the networks are not reduced, the proposed model cannot detect the attacks more accurately, thus reducing the system's performance. Therefore, an effective optimization algorithm is required to reduce the objective functions. The objective function is mathematically represented as,

$$O_f = min(L(P_d, e_d) + W_t) \tag{30}$$

The mathematical model of POA is mentioned below.

Mathematical model for POA: POA is a population-based algorithm that considers the population as a candidate solution. Therefore, the pelican population or candidate solution is randomly initialized based on the lower and upper bound and is mathematically given as,

$$p_{x,y} = L_y + rand\big(U_y - L_y\big), x = 1,2,\ldots.,m, y = 1,2,\ldots..,n, \tag{31}$$

The candidate solution is represented in the form of a matrix known as the population matrix and is given as,

$$P = \begin{bmatrix} P_1 \\ \vdots \\ P_x \\ \vdots \\ P_m \end{bmatrix}_{m\times n} = \begin{bmatrix} p_{1,1}\cdots p_{1,y}\cdots p_{1,n} \\ \vdots \quad \vdots \quad \vdots \\ p_{x,1}\cdots p_{x,y}\cdots p_{x,n} \\ \vdots \quad \vdots \quad \vdots \\ p_{m,1}\cdots p_{m,y}\cdots p_{m,n} \end{bmatrix}_{m\times n} \tag{32}$$

Here, the population matrix is denoted as $P$and the x^{th}pelican is denoted asP_x. The POA undergoes two stages in reducing the objective functions. The two stages are exploitation and exploration. The exploitation phase consists of the candidate moving towards the food source, and the exploration phase is the winging on the water surface.

Exploration stage: The exploration stage is considered as the candidate approaching the food source. The candidate approaching the food source is mathematically represented as,

$$p_{x,y}^{q_1} = \begin{cases} p_{x,y} + rand.\big(q_y - B.p_{x,y}\big)O_f < O_{fx}; \\ p_{x,y} + rand.\big(p_{x,y} - q_y\big), else, \end{cases} \tag{33}$$

Here, based on stage 1, the new progress of the x^{th}pelican in the y^{th}dimension is denoted as$p_{x,y}^{q_1}$, The location of the prey in the y^{th} dimension is denoted as q_y. Effective updating is done in the algorithm in which the candidate movement avoids entering the non-optimal regions and is mathematically given as,

$$P_x = \begin{cases} P_x^{q_1}, {O_f}_x^{q_1} < O_{fx}; \\ P_x, else \end{cases} \tag{34}$$

Table 3. Pseudocode for the proposed methodology

```
Begin
Pre-processing
Z-score normalization
    Return normalized value based on $\mu$ and $\sigma$ using equation 1
Feature selection
HPS-WOA
    Initialize parameters
    Execute WOA
    Set the resulted WOA as the initial point $a_0$ for PS
    Generate mesh points
    Compute objective function for each mesh point using equation 10
      if $O_f < I_p$
        $a_1 \times 2$ (Expansion factor)
      else
        $a_1 \times 0.5$ (Contraction factor)
    Repeat steps until termination criteria is reached
    print selected features
Attack detection and classification
Provide selected features to OptiBiNet_GRU
    Initialize four layers for classification
    Input layer: selected features
    Bi-LSTM and GRU to achieve the best accuracy in classification
    Dense layer: classification
    Compute loss function using equation 29
Hyperparameter tuning
Pelican Optimization Algorithm
    Initialize pelican population based on lower and upper bounds using equation 31
    for each pelican member
    Compute objective function using equation 30
        Minimize loss function using equation 34
        Minimize network weight using equation 36
Stop
```

Table 4. Statistics of attack and normal data records of the UNSW-NB15 dataset

Label	Testing dataset	Training dataset	Validation dataset
Attack data	45,071	157,748	22,535
Normal data	289,777	1,014,221	144,889

Exploitation stage: The exploitation stage is considered the stage of winging on the surface. In this phase, the pelican spread their wings under the water and moved the fish upwards to gather the food in its throat pouch. The mathematical expression of the pelican during hunting is given as,

$$p_{x,y}^{q_2} = p_{x,y} + A.\left(1 - \frac{i}{I}\right).(2.rand - 1).p_{x,y}, \quad (35)$$

The new pelican location is accepted or rejected based on an effective updating process and is mathematically represented as,

$$P_x = \begin{cases} P_x^{q_2}, O_{f_x}^{q_2} < O_{f_x} \\ P_x, else, \end{cases} \quad (36)$$

The pseudocode for the proposed methodology is shown in Table 3.

4. Results and discussion

The section briefly describes the two datasets used to examine the method's efficiency and the performance metrics used to evaluate the method. A separate performance analysis is given for both datasets.

4.1 Dataset description

UNSW-NB15 dataset: The cyber security research group created the UNSW-NB15 dataset at the australian centre for cyber security (ACCS). The dataset consists of 42 attributes, 44 features and 2 classes. The dataset contains 9 attack types: Fuzzers, analysis, backdoor, DoS, exploits, generic, reconnaissance, shellcode, and worms.

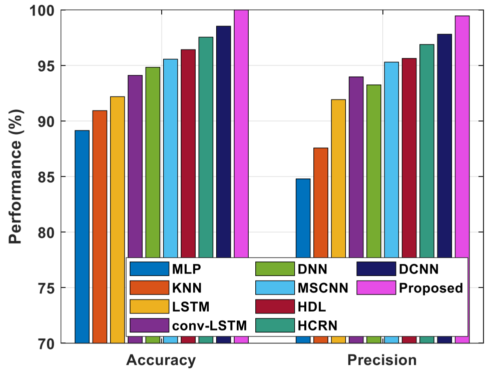
The statistics of attack and normal data instances for testing, training, and validation sets of the UNSW-NB15 dataset are shown in Table 4. The testing dataset comprises 45,071 attacks and 289,777 normal testing data records. The training dataset comprises 157,748 attacks and 1,014,221 normal training data records. The validation dataset is composed of 22,535 attacks and 144,889 normal data records.

CIC-IDS2017 dataset: The CIC-IDS2017 dataset was created by the canadian institute for cybersecurity (CIC). The dataset is composed of 14 attacks types: DoS goldenEye, heartbleed, DoS hulk, DoS slowhttp, DoS slowloris, SSH-patator, FTPPatator, web attack-brute force, web attack-SQL injection, web attack-XSS, infiltration, bot, PortScan, and DDoS.

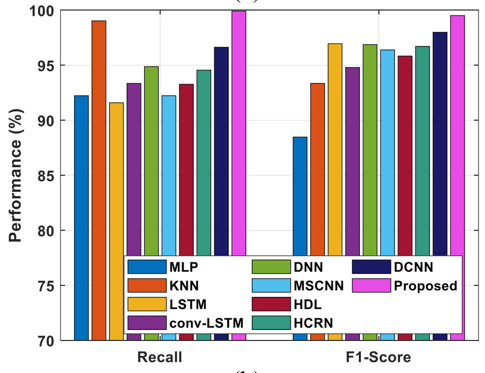
The statistics of attack and normal data instances for testing, training, and validation sets of the CIC-IDS2017 dataset are shown in Table 5. The testing dataset comprises 2229 attacks and 90,861 normal testing data records. The training dataset comprises

Table 5. Statistics of attack and normal data records of the CIC-IDS2017 dataset

Label	Testing dataset	Training dataset	Validation dataset
Attack data	2229	7800	1114
Normal data	90,861	318,014	45,431



(a)



(b)

Figure. 6: (a) Accuracy and precision and (b) recall and f1-score for UNSW-NB15 dataset

7800 attacks and 318,014 normal training data records. The validation dataset is composed of 1114 attacks and 45,431 normal data records.

4.2 Performance metrics

Intrusion detection is considered an important module in the proposed model, so the model's efficiency needs to be evaluated to determine the classification accuracy. The four performance metrics, accuracy, precision, f1-score, and recall, are the most widely used to validate intrusion detection systems' performance. A brief description of the performance metrics utilized is given below.

Accuracy: Accuracy is defined as the ratio of precisely classified samples to the total number of samples that depicts the overall performance of the proposed model and is mathematically represented as,

$$A_c = \frac{t^p + t^n}{t^p + t^n + f^p + f^n} \tag{37}$$

Here, true positive is denoted as t^p, true negative is denoted as t^n false positive as f^p and false negative as f^n.

Precision: It is defined as the ratio of the count of samples perfectly determined as a category to the count of samples determined and is mathematically represented as,

$$P_r = \frac{t^p}{t^p + f^p} \tag{38}$$

F1-score: It is referred to as the harmonic mean of recall and precision and is mathematically represented as,

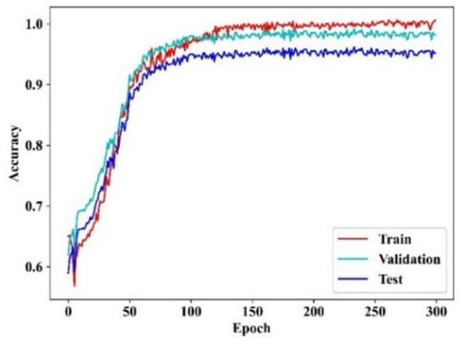
$$F1_s = \frac{2 \times P_r \times R_e}{P_r + R_e} \tag{39}$$

Recall: Recall is also known as the detection rate (DR) because the number of samples in a given class is closely related to the actual number of samples in that class. It is represented mathematically as,

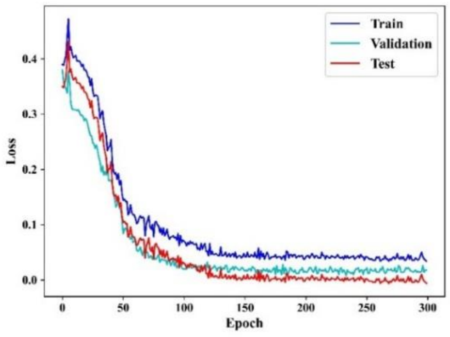
$$R_e = \frac{t^p}{t^p + f^n} \tag{40}$$

4.3 Performance analysis on the UNSW-NB15 dataset

The performance metrics such as accuracy, precision, recall, and f1-score are evaluated for the proposed method with some existing methods. The proposed method is compared with different existing baselines such as HDL [24], HCRN [25], DCNN [26], KNN (K-nearest neighbour) [28], multi-scale convolutional neural network (MSCNN) [29], MLP [30], LSTM [31], Conv-LSTM (convolution long-short term memory) [32], and DNN (deep neural network) [33]. Fig. 6 (a) shows the accuracy and precision rates of the proposed and existing methods for the UNSW-NB15 dataset. Fig. 6 (b) shows the recall and f1-score for the proposed and existing methods for the UNSW-NB 15 dataset. Comparing all the performance metrics, the proposed method is at a high peak for the entire performance metrics with an accuracy of 99%, precision of 95%, f1-score of 97%, and recall of 98%. This is because the proposed method used effective hybrid classification DL models and is used to reduce its loss and weight optimization algorithm. The classification accuracy of different classes using the proposed classifier for the UNSW-NB15 dataset is normal of 99%, exploits



(a)



(b)

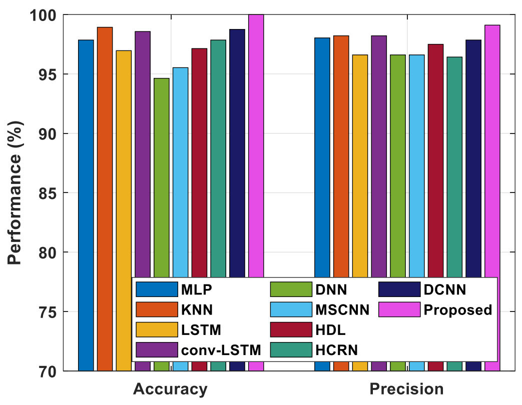
Figure. 7: (a) Accuracy curve and (b) loss curve for UNSW-NB15 dataset

of 99%, reconnaissance of 99%, DoS of 99%, generic of 99%, shellcode of 99%, fuzzers of 99%, worms of 100%, backdoors of 99%, and analysis of 99%. Therefore, the overall detection rate for the proposed method is 98%, and the false alarm rate is 0.04% for the UNSW-NB15 dataset.

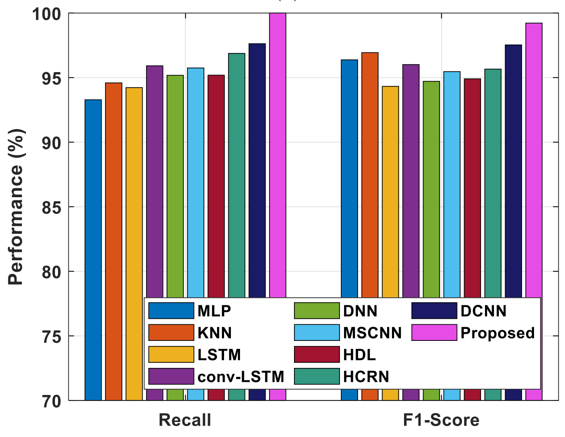
Fig. 7 (a) shows the accuracy curve, and Fig. 7 (b) shows the loss curve for the UNSW-NB15 dataset. The experiment with 300 epochs showed the best validation and training accuracy. After 300 epochs, the performance of validation and training declined. This is due to overfitting, and to overcome this, the experiment is terminated at 50 epochs. The proposed method in terms of validation, testing and training accuracy is shown in Fig. 7 (a). The testing, training, and validation loss are shown in Fig. 7 (b). The testing, training, and validation in accuracy graph is increased, but in loss graph the testing, training, and validation are decreased for the UNSW-NB15 dataset.

4.4 Performance analysis on the CIC-IDS2017 dataset

The performance metrics such as accuracy, precision, recall, and f1-score are evaluated for the proposed method with some existing baselines. Fig. 8 (a) shows the accuracy and precision rates of the proposed and existing methods for the CIC-IDS2017 dataset. Fig. 8 (b) shows the recall and f1-score for the proposed and existing methods for the CIC-



(a)

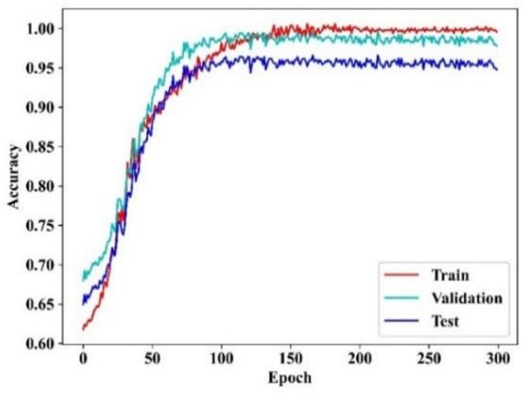


(b)

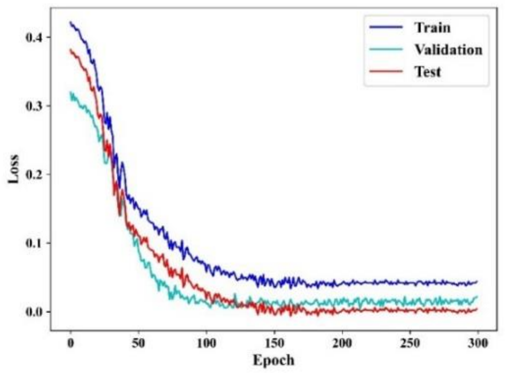
Figure. 8: (a) Accuracy and precision and (b) recall and f1-score for CIC-IDS2017 dataset

IDS2017 dataset. Comparing all performance metrics, the proposed method achieves the best performance for all performance metrics with an accuracy of 99%, precision of 99%, f1-score of 99%, and recall of 99%. The classification accuracy of different classes using the proposed classifier for the CIC-IDS2017 dataset is benign of 99%, DDoS of 99%, portscan of 99%, bot of 99%, infiltration of 100%, DoS slowloris of 99%, DoS slowhttptest of 99%, DoS hulk of 99%, DoS goldeneye of 99%, and heartbleed of 100%. Therefore, the overall detection rate for the proposed method is 99%, and the false alarm rate is 0.002% for the CIC-IDS2017 dataset.

Fig. 9 (a) shows the accuracy curve, and Fig. 9 (b) shows the loss curve for the CIC-IDS2017 dataset. The experiment with 300 epochs showed the best validation and training accuracy. After 300 epochs, the performance of validation and training declined. The proposed method in terms of validation, testing and training accuracy is shown in Fig. 9 (a). The testing, training, and validation loss are shown in Fig. 9 (b). The testing, training, and validation in accuracy graph is increased but in loss graph the testing, training, and validation are decreased for the CIC-IDS2017 dataset.



(a)



(b)

Figure. 9: (a) Accuracy curve and (b) loss curve for CIC-IDS2017 dataset

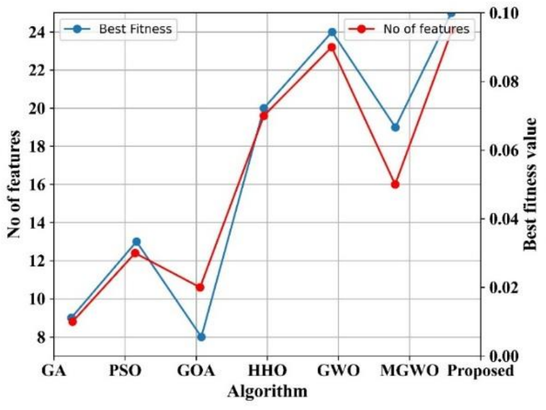
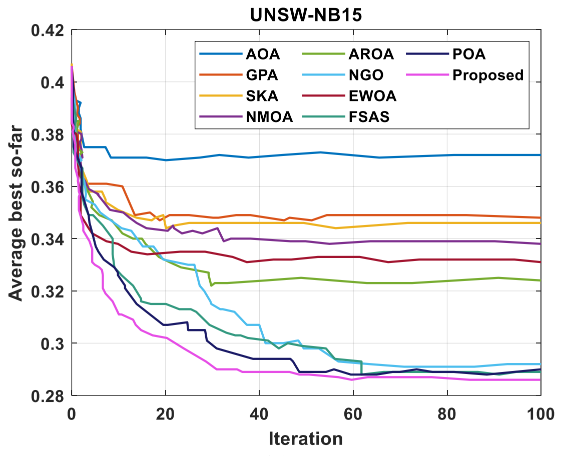


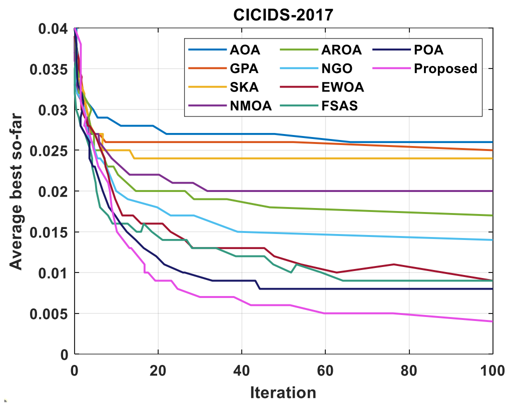
Figure. 10 Fitness and selected features

4.5 Performance analysis of feature selection technique

A hybrid feature selection technique is used in which its performance is evaluated based on the performance metric fitness and the selected features as shown in Fig. 10. The proposed optimization algorithm is compared with various existing optimization algorithms such as GA (Genetic algorithm), PSO (Particle swarm optimization), GOA (Grasshopper optimization algorithm), HHO (Horse herd optimization), GWO (Grey wolf optimization), and MGWO (Modified GWO). The proposed method



(a)



(b)

Figure. 11 Convergence curve: (a) Convergence curve for UNSW-NB15 dataset and (b) Convergence curve for the CICIDS-2017 dataset

uses a hybrid optimization algorithm for feature selection. As a result, the proposed optimization attains the best fitness value for greater selected features than other existing optimization algorithms.

Fig. 11 shows the convergence curve with the proposed and existing optimization algorithms such as Aquila optimization algorithm (AOA) [34], guided pelican algorithm (GPA) [35], stochastic komodo algorithm (SKA) [36], modified moth flame optimization algorithm (NMOA) [37], artificial rabbits optimization algorithm (AROA) [38], northern goshawk optimization (NGO) [39], enhanced whale optimization algorithm (EWOA) [40], fixed step average and subtraction (FSAS) [41] and puzzle optimization algorithm (POA) [42]. Fig. 11 (a) shows the convergence curve for the UNSW-NB15 dataset, and Fig. 11 (b) shows the convergence curve for the CICIDS-2017 dataset. From the figure, it is analyzed that the proposed optimization algorithm initially and gradually converges, that reveals its exploration capacity and finally reaching its optimum value. Furthermore, if an algorithm convergence towards the optimum, then the algorithm is said to be comparatively faster than other algorithms. As a result, the proposed optimization algorithm outperforms other existing meta-heuristic algorithms. The results show that the proposed method can more accurately detect and classify network attacks.

5. Conclusion

The IDS plays a significant role in discovering malicious activities and provides better network security solutions than conventional defence techniques, such as firewalls. With the help of ML techniques, such systems can determine attacks more accurately by detecting the relevant data patterns. However, the nature of the network data, the time-varying environment, and the unknown occurrence of attacks made the learning task very complex. The new attacks are difficult to monitor due to their new features and pattern types. So, the proposed work uses the DL technique to detect and classify network attacks. The work undergoes three stages: pre-processing, feature selection, and classification. These stages are well performed using various effective techniques and optimization algorithms. The method's effectiveness is validated using real-time datasets with various performance metrics. Performance analysis is done separately for classification and optimization techniques. The proposed method outperforms other methods in terms of accuracy of 99%, precision of 95%, f1-score of 97%, specificity of 99%, and recall of 98% for UNSW-NB15 dataset, whereas accuracy of 99%, precision of 95%, f1-score by 99%, specificity of 99%, and recall of 99% for CICIDS 2017 dataset, respectively. These results showed that the proposed method could detect and classify network attacks more accurately. The real-time applications in which IDS is used are network traffic processing, threat reporting, threat classification, signature matching, and anomaly detection. In our future work, the deep learning-based feature extraction (DLFE) and optimal robust pattern matching (ORPM) solution will be introduced for detecting unknown and malicious activities in networking systems.

Conflicts of interest

Authors have no conflict of interest to declare.

Author contributions

D. Shankar: Writing - Original Draft, Preparation, specifically writing the initial draft (including substantive translation).

G.Victo Sudha George: Writing - Review & Editing Preparation, specifically critical review, commentary or revision.

N kanya:Writing - Review & EditingPreparation, specifically critical review, commentary or revision.

Nomenclature

Symbols	Explanations
$a_{normalize}$	normalized value
a_i	data sample
μ	mean vector of features
σ	standard deviation
t	current iteration
$\vec{R}$ and $\vec{Q}$	coefficient vectors
A^*	best solution
$\vec{A}$	position vector
$\left\| \; \right\|$	absolute value
$\cdot$	multiplication
$\vec{s}$	random vectors
C	Constant [-1, 1]
r	random number
b	Random number [0, 1]
$\bar{A}_{rand}$	random position vector
f_v	Feature vector
t_s	Time step
E_{t_s}	input data
$\overrightarrow{h}_{s-1}$	past hidden state values
$\overrightarrow{h}_{s}$	hidden state values
E_{t_s}	present input data sequence
$\overleftarrow{h}_{s+1}$	future hidden state value
rg_{t_s}	gate vector
O_{t_s}	sigmoid function
Z_{t_s}	Output data
p_d	Detected probability distribution
e_d	true probability distribution
$L(P_d, e_d)$	Loss function
W_t	weight function
$p_{x,y}$	candidate solution of y^{th}variable
$mandn$	problem variables
$rand$	Random number [0, 1]
L_y	Lower bound
U_y	Upper bound
P	population matrix
O_f	objective function
B	Random number [0 or 1]
$P_x^{q_1}$	New state of pelican
$O_{f_x}^{q_1}$	Objective function based on stage 1
$p_{x,y}^{q_2}$	pelican's new status in stage 2
A	Constant (0.2)
$A \cdot \left(1 - \frac{i}{I}\right)$	neighbourhood radius
$P_x^{q_2}$	x^{th}pelican's new status
$O_{f_x}^{q_2}$	objective function based on stage 2

References

[1] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study", *Journal of Information Security and Applications*, Vol. 50, p. 102419, 2020.

[2] D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti, "A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems", *IEEE Systems Journal*, Vol. 15, No. 2, pp. 1717-1731, 2020.

[3] M. A. Ferrag, L. Maglaras, H. Janicke, and R. Smith, "Deep learning techniques for cyber security intrusion detection: A detailed analysis", In: *Proc. of 6th International Symposium for ICS and SCADA Cyber Security Research 2019*, Vol. 6, pp. 126-136, 2019, September.

[4] A. A. Salih and A. M. Abdulazeez, "Evaluation of classification algorithms for intrusion detection system: A review", *Journal of Soft Computing and Data Mining*, Vol. 2, No. 1, pp. 31-40, 2021.

[5] M. O. Okay, R. Samet, O. Aslan, and D. Gupta, "A Comprehensive Systematic Literature Review on Intrusion Detection Systems", *IEEE Access*, Vol. 9, pp.157727-157760, 2021.

[6] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, "Modeling realistic adversarial attacks against network intrusion detection systems", *Digital Threats*: *Research and Practice* (*DTRAP*), Vol. 3, No. 3, pp. 1-19, 2021.

[7] B. Subba and P. Gupta, "A tfidfvectorizer and singular value decomposition based host intrusion detection system framework for detecting anomalous system processes", *Computers and Security*, Vol. 100, p. 102084, 2021.

[8] I. A. Turaiki and N. Altwaijry, "A convolutional neural network for improved anomaly-based network intrusion detection", *Big Data*, Vol. 9, No. 3, pp. 233-252, 2021.

[9] M. Rashid, J. Kamruzzaman, T. Imam, S. Wibowo, and S. Gordon, "A tree-based stacking ensemble technique with feature selection for network intrusion detection", *Applied Intelligence*, pp. 1-14, 2022.

[10] M. A. Omari, M. Rawashdeh, F. Qutaishat, H. M. Alshira, and N. Ababneh, "An intelligent tree-based intrusion detection model for cyber security", *Journal of Network and Systems Management*, Vol. 29, No. 2, pp. 1-18, 2021.

[11] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection", In: *Proc. of 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1222-1228, 2017, September IEEE.

[12] Y. Lin, J. Wang, Y. Tu, L. Chen, and Z. Dou, "Time-related network intrusion detection model: a deep learning method", In: *Proc. of 2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1-6, 2019, December. IEEE.

[13] R. K. Vigneswaran, R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security", In: *Proc. of 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1-6, 2018, July. IEEE.

[14] P. V. Pramila and M. Gayathri, "Analysis of Accuracy in Anomaly Detection of Intrusion Detection System Using Naïve Bayes Algorithm Compared Over Gaussian Model", *ECS Transactions*, Vol. 107, No. 1, p. 13977, 2022.

[15] K. Hughes, K. McLaughlin, and S. Sezer, "A Model-Free Approach to Intrusion Response Systems", *Journal of Information Security and Applications*, Vol. 66, p. 103150, 2022.

[16] K. A. Tait, J. S. Khan, F. Alqahtani, A. A. Shah, F. A. Khan, M. U. Rehman, and J. Ahmad, "Intrusion detection using machine learning techniques: an experimental comparison", In: *Proc. of 2021 International Congress of Advanced Technology and Engineering (ICOTEN)*, pp. 1-10, 2021, July. IEEE.

[17] Y. Uhm and W. Pak, "Service-aware two-level partitioning for machine learning-based network intrusion detection with high performance and high scalability", *IEEE Access*, Vol. 9, pp. 6608-6622, 2021.

[18] S. W. Lee, M. Mohammadi, S. Rashidi, A. M. Rahmani, M. Masdari, and M. Hosseinzadeh, "Towards secure intrusion detection systems using deep learning techniques: Comprehensive analysis and review", *Journal of Network and Computer Applications*, Vol. 187, p. 103111, 2021.

[19] S. Patidar and I. S. Bains, "Intrusion Detection Using Deep Learning", *Inventive Computation and Information Technologies*, pp. 113-125, 2021.

[20] R. Lohiya and A. Thakkar, "Intrusion detection using deep neural network with antirectifier layer", *Applied Soft Computing and Communication Networks*, pp. 89-105, 2021.

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

[22] V. Ravi, R. Chaganti, and M. Alazab, “Recurrent deep learning-based feature fusion ensemble meta-classifier approach for intelligent network intrusion detection system”, *Computers and Electrical Engineering*, Vol. 102, p. 108156, 2022.

[23] M. Rani, “Effective network intrusion detection by addressing class imbalance with deep neural networks multimedia tools and applications”, *Multimedia Tools and Applications*, Vol. 81, No. 6, pp. 8499-8518, 2022.

[24] M. D. Moizuddin and M. V. Jose, “A bio-inspired hybrid deep learning model for network intrusion detection”, *Knowledge-Based Systems*, Vol. 238, p. 107894, 2022.

[25] M. A. Khan, “HCRNNIDS: hybrid convolutional recurrent neural network-based network intrusion detection system”, *Processes*, Vol. 9, No. 5, p. 834, 2021.

[26] R. V. Mendonça, A. A. Teodoro, R. L. Rosa, M. Saadi, D. C. Melgarejo, P. H. Nardelli, and D. Z. Rodríguez, “Intrusion detection system based on fast hierarchical deep convolutional neural network”, *IEEE Access*, Vol. 9 pp. 61024-61034, 2021.

[27] J. Nasiri and F. M. Khiyabani, "A whale optimization algorithm (WOA) approach for clustering", *Cogent Mathematics & Statistics*, Vol. 5, No. 1, p. 1483565, 2018.

[28] L. Xiong and Y. Yao, "Study on an adaptive thermal comfort model with K-nearest-neighbors (KNN) algorithm", *Building and Environment*, Vol. 202, p. 108026, 2021.

[29] Q. Fu, S. Li, and X. Wang, "Mscnn-am: a multi-scale convolutional neural network with attention mechanisms for retinal vessel segmentation", *IEEE Access*, Vol. 8, pp. 163926-163936, 2020.

[30] M. Desai and M. Shah, "An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (MLP) and Convolutional neural network (CNN)", *Clinical eHealth*, Vol. 4, pp. 1-11, 2021.

[31] S. Mekruksavanich and A. Jitpattanakul, "Lstm networks using smartphone data for sensor-based human activity recognition in smart homes", *Sensors*, Vol. 21, No. 5, p. 1636, 2021.

[32] M. Moishin, R. C. Deo, R. Prasad, N. Raj, and S. Abdulla, "Designing deep-based learning flood forecast model with ConvLSTM hybrid algorithm", *IEEE Access*, Vol. 9, pp. 50982-50993, 2021.

[33] D. Bau, J. Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, and A. Torralba, "Understanding the role of individual units in a deep neural network", In: *Proc of the National Academy of Sciences*, Vol. 117, No. 48, pp. 30071-30078, 2020.

[34] C. H. N. Kumari et al., “Aquila Optimizer Based Optimal Allocation of Soft Open Points for Multi Objective Operation in Electric Vehicles Integrated Active Distribution Networks”, *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, 2022, DOI: 10.22266/ijies2022.0831.25.

[35] P. D. Kusuma et al., “Guided Pelican Algorithm”, *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 6, 2022, DOI: 10.22266/ijies2022.1231.18.

[36] P. D. Kusuma et al., “Stochastic Komodo Algorithm”, *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, 2022, DOI: 10.22266/ijies2022.0831.15.

[37] S. Ummadisetty, et al., “Automatic Atrial Fibrillation Detection Using Modified Moth Flame Optimization Algorithm”, *International Journal of Intelligent Engineering and Systems*, Vol. 16, No. 1, 2023, DOI: 10.22266/ijies2023.0228.38.

[38] C. R. Rao et al., “Artificial Rabbits Optimization Based Optimal Allocation of Solar Photovoltaic Systems and Passive Power Filters in Radial Distribution Network for Power Quality Improvement”, *International Journal of Intelligent Engineering and Systems*, Vol. 16, No. 1, 2023, DOI: 10.22266/ijies2023.0228.09.

[39] M. Manohara et al., “Northern Goshawk Optimization for Optimal Allocation of Multiple Types of Active and Reactive Power Distribution Generation in Radial Distribution Systems for Techno-Environmental Benefits”, *International Journal of Intelligent Engineering and Systems*, Vol. 16, No. 1, 2023, DOI: 10.22266/ijies2023.0228.08.

[40] N. D. S. S. K. Relangi et al., “Effective Groundwater Quality Classification Using Enhanced Whale Optimization Algorithm with Ensemble Classifier”, *International Journal of Intelligent Engineering and Systems*, Vol. 16, No. 1, 2023, DOI: 10.22266/ijies2023.0228.19.

[41] P. D. Kusuma et al., “Fixed Step Average and Subtraction Based Optimizer”, *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, 2022, DOI: 10.22266/ijies2022.0831.31.

[42] F. A. Zeidabadi et al., “POA: Puzzle

Optimization Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 1, 2022, DOI: 10.22266/ijies2022.0228.25.