Networks and Heterogeneous Media

https://www.aimspress.com/journal/nhm

NHM, 20(2): 532–565.
DOI: 10.3934/nhm.2025024
Received: 15 October 2024
Revised: 14 April 2025
Accepted: 27 April 2025
Published: 27 May 2025

Research article

A novel improved deep learning model based on Bi-LSTM algorithm for intrusion detection in WSN

Ra'ed M. Al-Khatib[1,*], Laila Heilat[1], Wala Qudah[1], Salem Alhatamleh[1] and Asef Al-Khateeb[2]

[1] Department of Computer Sciences, Faculty of Information Technology and Computer Science, Yarmouk University, Irbid 21163, Jordan

[2] Department of Computer Science, Faculty of Computer Science and Information Technology, Jerash, Jordan

* **Correspondence:** Email: raed.m.alkhatib@yu.edu.jo.

Abstract: The widespread range of wireless sensor networks (WSNs) has recently increased the possibility of attacks. In the cyber-physical system, WSN is considered a significant component, consisting of several hops that self-organize from moving or stationary sensors. Attackers perform abusive operations to enter, seize, and control the WSN network. In order to stop such attacks on sensor networks in the future, the network traffic data was evaluated, dangerous traffic activity was monitored, and nodes were investigated. In this study, we proposed a new improved bidirectional long-short-term memory (Bi-LSTM) algorithm, which is an enhancement of the LSTM model, to address the issue of intrusion detection in WSN systems, which has four steps. In a preprocessing step, the initial raw data collected from the network can be used to train learning models. Then, the analyzed data was classified according to the types of network traffic, and the attacks were detected in the second step. In the next step, our proposed improved learning models showed results with higher accuracy than traditional detection methods. This study assessed the performance of deep learning algorithms on two different datasets, primarily the 'WSN-DS' and 'KDDCup99' datasets, which were chosen to identify and classify different DoS attacks. The primary common WSN attacks were flooding, scheduling, blackhole, and grayhole, which were included in the data set 'WSN-DS', and (denial-of-service (DoS), user-2-root (U2R), root-2-local (R2L), and Probe) were in the 'KDDCup99' dataset. Our newly improved proposed model based on Bi-LSTM achieved an accuracy of 100% on the 'WSN-DS' dataset and 99.9% on the 'KDDCup99' dataset.

Keywords: artificial intelligence; recurrent neural network (RNN); deep learning; intrusion detection; wireless sensor network (WSN); bidirectional long-short-term memory (Bi-LSTM)

1. Introduction

The wireless sensor networks (WSNs) recently gained great importance and became a critical research field due to their various real-time applications, including the healthcare industry, building security control, critical military monitoring, and monitoring of jungle fires [1, 2]. Figure 1 illustrates the general framework of the WSN, which consists of several dispersed nodes organized into several clusters. The cluster head (CH) is a single master sensor node that is then remotely connected to the sensor nodes in each cluster. This CH node compiles the information gathered from every sensor node within each group to provide information to the base station. Therefore, the data may be vulnerable to hacking or Denial-of-Service attacks during this final processing or in the analysis stage. Given this widespread nature, it has become necessary to protect data sent over the internet and maintain the network through intrusion detection systems (IDS). IDS is a mechanism similar to wireless sensor networks, as it constantly monitors data movement and determines whether it is normal or has been attacked. In the case of WSN and Bluetooth, authentication techniques and security protocols cannot be used due to limited resources [3].

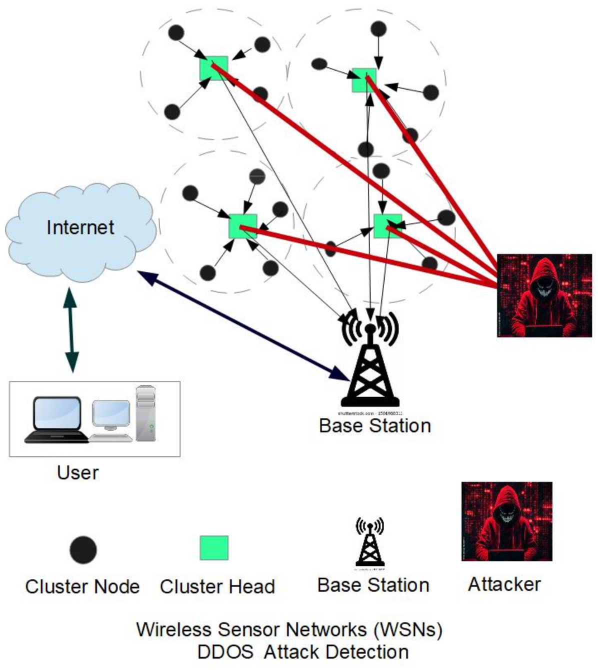

Figure 1. General framework for WSN structure, some parts adapted from [2].

There are many challenges, but the main challenge is to protect the WSN. Using traditional methods, such as encryption, is difficult due to limited processing capabilities [4]. However, most existing systems fail to classify adverse events more accurately. The best number of features from the dataset can be used to categorize after the features have been chosen [5]. Feature selection dynamically adjusts the number of chosen features using various methods. Consequently, the underlying challenge of categorizing network data requires the creation of intelligent and efficient methods that can recognize security breaches through behavior and network traffic analysis [6, 7]. To solve the aforementioned issues, we proposed a new model based on improving the bidirectional long-short-term memory (Bi-LSTM) to build a powerful intrusion detection system in this study. Our new improved Bi-LSTM-based model is well-trained and evaluated using the 'WSN-DS' and 'KDDCup99' datasets.

The rest of this paper is as follows: Section 1 explains the essential ideas of the intrusion detection domain. Section 2 displays the previous works. Section 3 illustrates the background on IDS and bidirectional LSTM techniques. Section 4 discusses our development methodology based on enhancing the Bi-LSTM technique for tackling the IDS problem. Section 5 covers the findings from the experiments. Section 6 presents the conclusion with future remarks.

2. Related work

A growing collection of literature examines various elements of IDSs and WSNs. The significant interest in WSNs has recently driven research and practical applications in IDSs. A brief overview of several significant relevant works in the field is given in this section. This section will also provide a detailed outline of various important related works in the field of IDSs.

In [8], the authors discussed four community intrusion detection systems (CIDs) made for WSNs. These four CIDs are developed using genetic algorithms, neural networks, embedded systems, fuzzy neural networks, and classified support vector machine (SVM), which were compared in this study based on their respective methodologies [9]. They found that SVM had the lowest errors in the system, during both training and testing in the evaluation processes.

The authors in [10] presented a deep learning-based method for quickly detecting and handling network intrusions in WSNs. The main goals were minimizing resource overhead, attaining high accuracy, and enhancing intrusion detection efficacy. The suggested strategy was developed based on deep learning techniques, which is in line with the increasing interest in applying these approaches to the subject of WSN security.

The study in [11] introduced the presentation of a contrastive learning model for network intrusion detection information within a supervised prototype label embedding framework. Traffic features and prototypes with embedded labels are placed in a shared embedded space. The comparison of their distances serves as a classification criterion. The model was evaluated in both the NSL-KDD and UNSW-NB15 datasets, achieving 83.7% accuracy in NSL-KDD, and 89.9% accuracy in UNSW-NB15. It faced the challenge of class imbalances, thus compromising its efficiency in merely detecting the minority attack classes, and it badly needs optimization for broader applicability across other network environments.

The work in [12] investigated how to predict the connectivity of wireless internet of things (IoT) devices based on data on mobile device usage for 30 days on more than 6214 mobile devices. They applied machine learning techniques like AutoRegressive Integrated Moving-Average (ARIMA), ARIMA with eXogenous covariates (ARIMAX), Hidden Markov Models, Exponential Smoothing, Logistic Regression, Random Forest, and Gradient Boosting to the gathered data. It achieved the highest accuracy, with ARIMAX reaching up to 93%, followed by Random Forest, Logistic Regression, and over 90% for ARIMA. In contrast, ARIMAX consumed more computing time, leaving room for a simpler model to act. This research also indicated the need for efficient feature selection because there were soft patterns in the data. It also recommended applying clustering techniques for better prediction.

In [3], the researchers integrated many techniques to classify intrusion detection. It takes into account the unique characteristics and resource limitations of these networks. This proposal uses a likelihood vector machine (LSVM) and cuckoo search greedy optimization (CSGO). It locally combined linear embedding, genetic algorithm, oversampling process, SVM, and context to form the CSGO-LSVM model [13]. This technique produced the lowest error 11% ratio rate.

The primary objective in [4] was to assess how well various machine learning (ML) methods worked for WSN security. The proposal measured the efficiency of many intrusion detection algorithms. It focused on the importance of choosing the best algorithm to achieve WSN security, and provided good insight into the development and application of effective IDSs. Several ML algorithms were used on the 'KDDCup99' and 'WSN-DS' datasets. The Naïve Bayes algorithm showed high accuracy, as True Positive (TP) was approximately 92.9% for 'KDDCup99', and 95.3% for the 'WSN-DS' dataset. Likewise, the TP rate using Random Forest was between 99.7 and 100%. The final results were approximately 99.4–100% for both used datasets.

The outcome of [6] had a substantial impact on the WSN security sector. It presented a flexible and effective intrusion detection system that suits different operating conditions. It obtained high accuracy in intrusion detection and reduced the rate of false alarms. Compared to the classification accuracy, the model was better than some common classifiers, such as C4.5, decision tree (DT) classifiers with fuzzy temporal rules, SVMs, and multilayer perceptrons (MLP) classifiers. In five studies, it was tested on the skillful application of fuzzy temporal rules, resulting in improved predictions and a greater ability to interpret the system. Previous studies have explored various mathematical models to analyze network actions, including convergence analysis in graph-based systems. For instance, the work published in [14] has a comprehensive study that demonstrated almost second-order uniform accuracy in k-star graphs, validated through extensive numerical experiments.

Numerous studies have investigated various computational methods to examine network behavior and enhance intrusion detection systems in WSNs. For instance, in [15], numerical investigations have demonstrated the effectiveness of finite difference methods in solving complex fluid dynamics problems, particularly in simulating turbulent flows with high accuracy. This methodological robustness aligns with the need for precise modeling in network security, where accurate anomaly detection relies on advanced mathematical approaches. A recent study in [16] applied uniform convergence analysis to semi-linear systems with boundary and interior layers, in order to demonstrate a new improved computational efficiency and stability model. This aligns with our approach, where numerical stability plays a crucial role in optimizing the Bi-LSTM-based model for intrusion detection in WSNs.

The paper in [17] presented an intrusion detection system. It was specifically introduced for WSNs through gradient boosting machine (GBM) techniques with sequential backward selection (SBS) and the LightGBM method (SLGBM). Taking into consideration the particular difficulties presented by GBM, it is used by SLGBM to enhance WSN security. WSNs can benefit from using SLGBM as a good model system, and also an effective, flexible intrusion detection method for WSNs that was utilized in smart settings. The WSN-DS dataset was used for the test. The final findings demonstrated a high degree of accuracy in identifying the four assault types: Normal, Grayhole, Flooding, Blackhole, and Scheduling, where the attack rates are 99.8, 99.1, 96.5, 99.4, and 96.1%, respectively. It also indicates a clear improvement in the F-measure scores and timing scheduling of time division multiple access (TDMA) attacks.

The advanced numerical methods in [18], used to address time-delay interaction and propagation problems in the previous study, depend on improving the agreement between parameters. This work may contribute to enhancing the performance of smart algorithms, such as the Bi-LSTM model, in the field of intrusion detection in WSNs. Overall, it was a valuable study since the effectiveness of advanced neural networks in detection relies on the development of high-performance algorithms.

In [1], the authors introduced the 'WSN-DS' dataset, which is a valuable dataset for evaluating IDS in WSNs. It offers a unified standard with different stealth scenarios and network configurations. Thus, facilitating research and development to detect intrusion in WSN networks. A single hidden layer combined with a 10-fold cross-validation technique produced the greatest results. The classification accuracy for flood attacks was 99.4%, for black hole assaults was 92.8%, for gray hole attacks was 75.6%, for scheduling attacks was 92.2%, and for the typical scenario (no attacks) was an astonishing 99.8%. Table 1 provides an overview summary of previous research models.

Table 1. Summary of the related works.

Ref.	Method	Used dataset	Accuracy	Key findings and performance
[8]	CIDS in WSNs	Two hundred images of ten individuals	NA	Compared four Community IDSs (Embedded Systems, GA, ANN, SVM, Fuzzy Neural Network). SVM had the lowest errors during training and testing
[11]	Supervised contrastive learning with prototype-label embeddings	NSL-KDD and UNSW-NB15	83.7%, 89.9%	Improved classification performance but struggled with class imbalance, leading to lower detection rates for minority attack classes.
[12]	ARIMA, ARIMAX, HMM, ETS, Logistic Regression, Random Forest, GBM	6,214 mobile devices, 30 days of activity data	ARIMAX: 93% Others: >90%	ARIMAX achieved the highest accuracy but had a high computational cost. Random Forest and Logistic Regression provided similar accuracy with lower training time. No clear connectivity pattern was observed across the devices.
[3]	CSGO-LSVM for Enhanced Intrusion Detection	NSL-KDD and UNSW-B15	99.65%	Utilized CSGO-LSVM to enhance intrusion detection, achieving the lowest error rate
[4]	Machine Learning Algorithms for WSN Security	KDDCup99 and WSN-DS datasets	92.9% and 95.3%	Systematically compared machine learning algorithms in WSN security, highlighting the effectiveness of Naïve Bayes, IBK, and Random Forest
[6]	Adaptive Intrusion Detection in WSNs	a dataset called 'KDDCup99'	NA	Developed an effective intrusion detection system tailored to WSN challenges, improving accuracy and minimizing false alarms
[17]	SLGBM for WSN Intrusion Detection	WSN-DS dataset	Accuracy 99.8%	Proposed SLGBM as an efficient solution for intrusion detection in WSNs used in smart environments
[1]	WSN-DS Dataset for Evaluation	WSN-DS dataset	Accuracy 99.4% for Flooding attacks	Introduced the 'WSN-DS' dataset for testing and comparing IDSs in WSNs, providing a standardized dataset for experimentation

We were inspired by numerical methods using gradient mesh refinement in [19], to improve the accuracy and stability of our proposed model. The work in this [19] technique is mainly used to capture sharp changes in numerical solutions, similar to our efforts to extract fine-grained features from mesh data [20]. To sum up, we are inspired to improve the accuracy and stability of our model by numerical methods that use gradient mesh refinement.

3. Background

3.1. Intrusion detection system

The IDS aims to detect intrusion or harmful activities using security information that can respond when any intrusion occurs [3]. The most important IDS systems for intrusion come in two types, which are as follows:
1- Network intrusion detection systems, which have a function to analyze network traffic.
2- The other type is based on the host, which has a function to monitor files, and the main example is the necessary operating system.

This is important because it guarantees a high level of protection, particularly in modern work environments that require a high level of safety. Therefore, the IDS can be adapted to any attack, particularly because some attackers may use new attacks, such as intrusion detection systems. Frequently, it uses variables depending on the discovery of signs and the detection of anomalies. So, it depends on signature-based identification. The identifiers based on the signature depend on the potential being threatened to search for specific patterns, such as the byte sequence in the traffic of the network, or the well-known instruction sequence used by harmful programs. These terms arise from the antivirus program, which indicates that these patterns are signatures. Although signing knowledge can detect attacks that are easily known, it is impossible to discover new attacks. Additionally, identifying unknown attackers is not unusual. The anomaly-based attack depends on the deviation of modern technologies that are designed to detect any penetration of these attacks. It is unknown due to the spread of malicious software. So, some used methods are designing a model based on the activity of identifying and then comparing it with a reliable model to discover new types of attacks that can produce false positives.

3.2. Long-short-term memory

In 1997, LSTM was introduced as a deep learning (DL) model for the time domain [21]. It mainly differs from conventional neural networks (CNN) in that it includes an LSTM network with a memory unit and a forgetting gate in its hidden layer. During the training phase, the long-term information flow might happen to improve the neural network's memory capacity. Figure 2 illustrates the structure of the LSTM cell [21], which depicts four computational units: the memory unit, three gates for forgetting, input, and output.

Using the Sigmoid activation function, a new value f_t is created using the current layer input x_t, and the previous hidden layer output h_{t-1}. This function determines whether information learned from the previous cell state C_{t-1} will be remembered or forgotten. The modified cell state C_t is then saved.

The "Sigmoid" function, denoted as σ, has an output range of [0, 1]. Within LSTM, the f_t determines the degree to which long-term memory information is forgotten, acting as the forget gate's value within the range of [0, 1]. The b_f represents the bias of the forget gate, while W_f denotes the weight matrix associated with it. In LSTM, the threshold operation consists of an activation function called "Sigmoid", which is followed by a dot product operation. The information is updated or retained based on this method. This choice is made before the concealed layer from the previous time step enters the forget gate. On the other hand, information flow is facilitated by the constant horizontal shifts in the cell state. The equations for the function connecting h_{t-1}, x_t, and f_t is as follows:

$$I_t = \sigma(W_t.[h_{t-1}, x_t] + b_i), \tag{3.1}$$

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f), \tag{3.2}$$

$$C_t = C_{t-1} \times f_t + I_t \times \tilde{C}_t), \tag{3.3}$$

$$\tilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c)\ , \tag{3.4}$$

where b_c signifies the memory unit's tendency to be biased. C_t is the total of the updated state plus the initial state. I_t represents the value produced by the input gate. $\tilde{C}_t$ represents C_t transitory unit state. W_c shows the weight matrix of the memory unit. *tanh* refers to the hyperbolic tangent activation function.

$$h_t = tahn(C_t) * o_t, \tag{3.5}$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o), \tag{3.6}$$

where h_t is obtained through the "Sigmoid" layer. The b_o is the output gate's bias. The $tanh(C_t)$ ranges from -1 to 1. The o_t shows the value of output. The W_o represents the output gate's weight matrix.

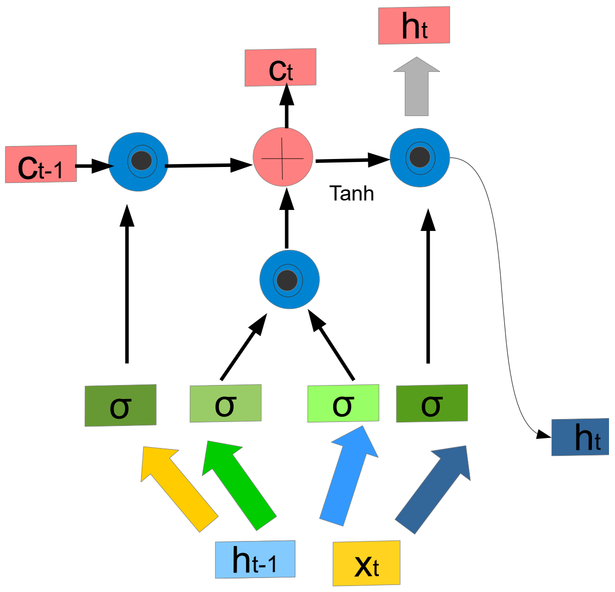

Figure 2. LSTM cell structure, adapted from [21].

After passing through the input, output, and forget gates, the signal helps retain and store information for the current time step. The neural network of the LSTM architecture demonstrates how quickly the input variable spreads from the input to the output. As a result, the model of forecasting prediction error gradually increases until it finally experiences a sharp spike, which is proportionate to the amount of time that has passed.

For power load forecasting over a single day, the cumulative LSTM prediction error is shown in Figure 3. For short-term load forecasting techniques, the training dataset typically lasts for one day or one week [22]. In Figure 3, three days' worth of data is utilized as an example of forecasting [23].

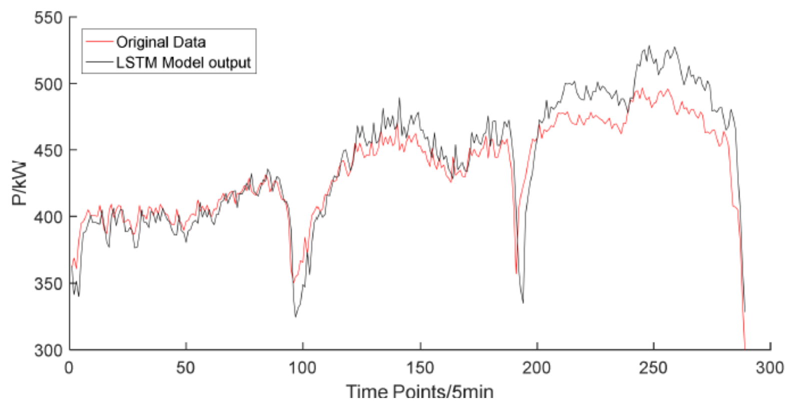

Figure 3. Rate accumulate error of the forecasting model, adapted from [21].

3.3. Bi-LSTM neural network

Bi-LSTM is proposed to solve the cumulative error issue. This Bi-LSTM includes two layers, and two methods are utilized to calculate the concealed vector: One calculates it from the front to the back, and the other from the rear to the front, determining the neural network's outputs as depicted in Figure 4.

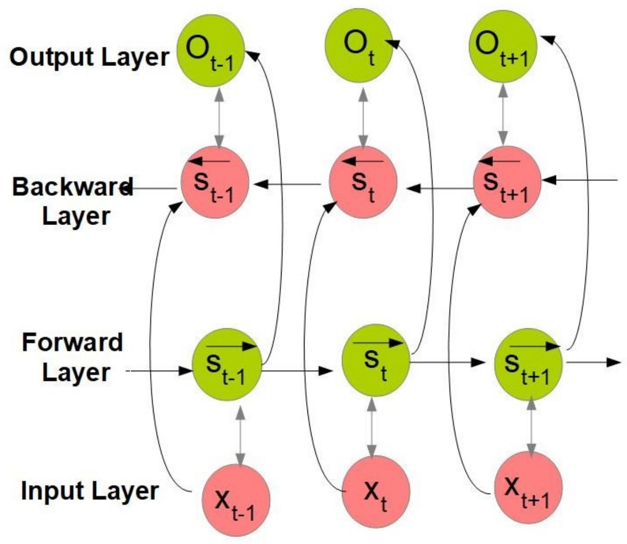

Figure 4. Basic structure of Bi-LSTM model, adapted from [21].

In the bidirectional function, the results are stored in the memory unit, and a directional loop links the layers with previous information. By integrating the past output with the neural network's present output, results are determined. Consequently, the volume of the input data in the series increases, and both gradient explosion issues and gradient problems dissipate.

In a bidirectional structure, prediction results are derived by considering both the last entry and the preceding entry to prevent the loss of relevant information. Figure 4 demonstrates that the front layer continually stores the output, which computes the forward direction from 1 to t.

Every time step, the output of the backward layer's calculation of the reverse time series is saved. The bi-LSTM combines the final outputs generated at each time point using the outputs of the forward and backward layers.

The following formulas explain how the bi-LSTM model works:

$$s'_t = f(u'x_t + W's'_{t+1}), \tag{3.7}$$

$$s_t = f(ux_t + Ws_{t-1}), \tag{3.8}$$

$$o_t = g(Vs_t + V's_t), \tag{3.9}$$

where

- s_t shows the hidden layer's state variable at a particular time t.
- x_t symbolizes the vector that was entered.
- g and f indicate roles of activation.
- s'_t represents the reverse hidden layer for the state variable fluctuating over time t.
- o_t shows the output layer's state variable at different times t.
- V', W', and U' show the corresponding reverse weight matrix.
- V, W, and U pertain to the weight matrices sent between the input layer and the hidden layer as well as between the hidden layer and the output layer.

Forward and backward layers do not share any information about the state weight matrix [23]. Every time, the outcome is determined by calculating the forward layer and the backward layer independently. The forward computation results are added to determine the final output o_t, the s_t, and the calculation of s'_t for the reverse hidden layer.

3.4. *Datasets*

We use two different types of datasets ('WSN-DS' and 'KDDCup99'). These used datasets have been prepared and designed by researchers in the literature as standard benchmarks to prepare comprehensive test scenarios for the attacks. i) The 'WSN-DS' is a dataset for intrusion detection systems in wireless sensor networks. The process of the data collection mechanism for 'WSN-DS' was explained by [1]. ii) The 'KDDCup99' was produced by the Advanced Research Projects Agency (DARPA), which is a data collection using network traffic obtained from [24, 25]. These two utilized types of datasets are presented and fully discussed in the subsequent sections.

3.4.1. Description of WSN-DS dataset

The wireless sensor network dataset ('WSN-DS') is basically collected and constructed from packets sent and received within a WSN, while the low-cost supervision service is required [1]. Also, extract the relevant information from sent and received packets in a wireless network, but we also have to make sure that the vital information about the network is collected in a way that facilitates the identification, classification, and eventual neutralization of any possible threats. Every sensor employed in this study will be involved in the observation process and must be able to continually observe a group of nearby nodes to share the workload among sensor nodes. Determining the optimal number of nodes and sensor nodes to monitor each network sensor effectively was the issue at hand. To determine this number, numerous experiments have been carried out, and a concise overview of the results can be seen in Table 2.

Table 2. Measurements for five simulation tests (A–E) to determine how many nodes and each node can monitor.

Number of neighbors to observe	The maximum number of monitors that a certain node can have					Prime number of monitors on a certain node					Total number of nodes under observation				
	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
3	6	7	7	6	7	0	0	0	1	0	97	99	99	100	97
4	7	9	8	8	9	0	0	0	1	0	99	99	99	100	99
5	10	9	10	10	10	1	1	1	1	2	100	100	100	100	100
6	11	12	11	10	13	1	1	1	2	2	100	100	100	100	100

One node can detect seven sensor nodes, and every sensor node detects three of its neighbors' nodes. Stated differently, the base station (BS) has been notified about the same node seven times by seven separate observing nodes. These reports could be examined for consistency to ensure that the information received is accurate. In certain cases, no sensor was keeping an eye on a certain sensor node. This suggests that to obtain information on every network sensor node, monitoring just three nearby nodes is insufficient.

Additionally, maintaining a check on four neighbors has improved things. However, in each of the five circumstances, all sensor nodes are only observed when the number is 5. When a sensor node saw six of its neighbors, similar outcomes were seen. As a result, research has revealed that keeping an eye on five neighbors is sufficient to get data about any node within the network. An additional observation would merely add computational complexity. The first step in the simulation is to select 5 neighbors to watch. Every node sends out a 'Hello' signal. Each node chooses the first five nodes, it hears from as a result. After that, it keeps an eye on them over the simulation time, ensuring that after each round, every node reports to its cluster head (CH). After that, the BS receives the reports from the CH. For security reasons, only one report from this node can be monitored, especially if there are suspicions about the CH. Moreover, if this node is located further away from the CH than from the BS, the reports can be sent directly to the electricity authority. However, this approach will result in higher energy consumption.

After a detailed analysis, we extracted 23 variables from the LEACH routing algorithm, which helps determine the condition of each network node. These extracted 23 attributes are listed in the following order.

Features of WSN-DS Dataset:
Node ID: A special number that can be used to identify the sensor node at any time and in any round.
Time: Simulation duration.
Is it CH? This signal indicates if the node is either a regular node value = 1 or value = 0.
ID: in the CH current cycle.
RSSI: The nodes received signal strength and CH for the current round.
Range to CH: The present round-trip distance between the node and its CH.
Maximum distance to CH: The greatest separation between a cluster node and the CH.
The average distance between a cluster node and its CH is known as the average distance to the CH.
Present energy level: The node's power at the moment in this cycle.
Energy usage: The amount of energy utilized in the prior period was specified.
ADV_CH send: The quantity of broadcast messages sent to the nodes to advertise CH was referred to as the "ADV_CH send".
The amount of advertisement that is sent to the CH is received by the ADV_CH.
Count of join request messages sent by nodes to the CH in order to request a join.
Received the join request: The received request is to know the number of messages nodes send to the CH to join as members.
ADV_SCH send: The total amount of broadcast messages sent to the nodes to advance the TDMA schedule is known as the ADV_SCH send.
ADV_SCH received: The quantity of messages sent by CHS using TDMA scheduling.
Rank: The position of the node about the TDMA timetable.
Data sent: The quantity of data packets sent by the sensor to its CH communication system is known as Data sent.
Data Received: The total number of data packets collected throughout the CH discussion.
Data Sent to BS: The amount of information sent to the base station in the form of data packets. Division in the exchange of messages.
The distance between the two base stations and the communication channel is known as the CH and BS.
Send Code: Identifies code that the team transmits.
Attack kind: In the event that the node is not an attacker, its kind. There are five potential values for this class: Normal, Blackhole, Scheduling (TDMA), Flooding, and Grayhole [1].

3.4.2. Attack schema

The used dataset incorporates the four denial-of-service (DoS) attack types identified in the LEACH protocol: Flooding, Scheduling (TDMA), Blackhole, and Grayhole attacks. Every one of these attacks will be modeled in this section. The network terrain has been split up into ten sections to guarantee that the attacker nodes are distributed appropriately. Subsequently, the attacker ratios are dispersed at random throughout these locations following the simulation scenarios.

Algorithm 1 Blackhole attack model

```
$N \rightarrow$ refers to Network size
$SN \rightarrow$ refers to Sensor Node
$MN \rightarrow$ refers to Malicious Node
$CH \rightarrow$ refers to Cluster Head
$BS \rightarrow$ refers to Base Station
$CM \rightarrow$ refers to a member of the cluster
$NC \rightarrow$ refers to the list of cluster heads
$x \rightarrow$ indicates to Integer number between 0 and $N - 1$
$\forall SN_i,\ in\ range\ 0 < i \leqslant N$, Compute $T(SN_i)$ and random $r_{SN_i}$
IF $(r_{SN_i} < T(SN_i))$ THEN
    $SN_i = CH$
ELSE
    $SN_i = CM$
ENDIF
$\forall CH_j,\ where\ j \in NC$
{
$CH_j$ indicates to broadcast the advertisement message ($ADV\_CH$)
$x$ and $CM$s are coming with $CH_j$
$CH_j$ produces TDMA schedule
$x$ and $CM$s utilize a matching TDMA slot to send data to $CH_j$
}
IF $CH_j = MN$ THEN
    executes the attack by discarding every packet
ELSE
    delivers collected data to $BS$
ENDIF
```

3.4.3. Blackhole attack

One type of DoS attack known as a "Blackhole attack" is intercepting the CH signal at the start of the round to interfere with the LEACH protocol. Therefore, data packets that will be sent to the BS for any node entering CH must first be transmitted for the data packets to reach the BS. Adopting the role of CH, the Blackhole attacker continues to drop these packets, refusing to send them to the BS [26]. Algorithm 1 illustrates the Blackhole assault method. In the simulation environment, a Blackhole assault has been built by randomly introducing an attacker, where the intensities are (10, 30, and 50%) of the sensor nodes. All packets relayed through these attackers, while they pose as CHs, will be dropped as they travel to the BS.

3.4.4. Grayhole attack

Through a type of DoS assault called a Grayhole attack, an attacker can interfere with the LEACH protocol and act as a CH for nodes. Consequently, when the established CH receives data packets from other nodes, it may drop certain packets or parts of packets (selectively or randomly) to keep them from reaching the BS [27]. The Grayhole attack technique is displayed in Algorithm 2. Similar to the Blackhole attack, the Grayhole attack involves randomly injecting 10, 30, and 50% of the sensor nodes. Furthermore, a random decision is made regarding whether or not to forward a specific packet. However, depending on how sensitive the sensed data that the packet is carrying is, selective judgment can be made. A particular packet is made at random.

Algorithm 2 Grayhole attack model

```
$N \rightarrow$ refers to the size of the Network
$SN \rightarrow$ refers to the node for sensor
$MN \rightarrow$ refers to the Malicious Node
$CH \rightarrow$ refers to the Cluster Head
$BS \rightarrow$ refers to the Base Station
$CM \rightarrow$ refers to the member in the cluster
$NC \rightarrow$ refers to the list of the cluster heads
$x \rightarrow$ refers to the value between 0 and $N - 1$ (integer value)
$\forall SN_i,\ in\ range\ 0 < i \leqslant N$ , Calculate $T(SN_i)$ and random $r_{SN_i}$
IF $(r_{SN_i} < T(SN_i))$ THEN
    $SN_i = CH$
ELSE
    $SN_i = CM$
ENDIF
$\forall CH_j,\ where\ \ j \in NC$
{
$CH_j$ Transmit to all the advertisement message ($ADV\_CH$)
$x$ and $CMs$ will come together with $CH_j$
$CH_j$ makes a schedule for TDMA.
$x$ and $CMs$ transfer data to $CH_j$ during the relevant TDMA time slot
}
IF $CH = MN$ THEN
    drops certain packets to carry out the attack (selectively or randomly)
ELSE
    delivers compiled data to $BS$
ENDIF
```

3.4.5. Flooding attack

A flooding attack is a type of DoS assault in which the attacker alters the LEACH protocol in different ways. By using strong transmission strength to deliver a high number of advertising CH messages (ADV_CH), this study investigates the effects of flooding assaults. As a result, when sensors pick up a lot of ADV CH signals, they will require more energy and time to determine which CH to join. Furthermore, the attacker tries to fool the victim into choosing it as a CH in an attempt to drain their energy, particularly for nodes that are far away from it [28]. Algorithm 3 demonstrates the operation of the flooding attack algorithm. There are several methods in which flooding assaults have been implemented in the simulation environment. According to certain research, the attacker delivered 10

ADV CH communications, while other investigations show that they sent 50 CH ADV messages, or a number randomly selected between 10 and 50. Theoretically, there will be an increase in the number of ADV CH messages sent and received, and more energy consumption per round based on the ratios of multiple attackers.

Algorithm 3 Flooding attack model

```
$N \rightarrow$ refers to the size of the Network
$SN \rightarrow$ refers to the node for Sensor
$MN \rightarrow$ refers to the Malicious Node in the network
$CH \rightarrow$ refers to the Cluster Head
$BS \rightarrow$ refers to the Base Station
$CM \rightarrow$ refers to the member in the cluster
$NC \rightarrow$ refers to the list of the cluster heads
$x \rightarrow$ refers to the value between 0 and $N-1$ (Integer Value)
$\forall SN_i,\ in\ range\ 0 < i \leqslant N$, Calculate $T(SN_i)$ and random $r_{SN_i}$
IF   $(r_{SN_i} < T(SN_i))$   THEN
    $SN_i = CH$
ELSE
    $SN_i = CM$
ENDIF
$\forall\ CH_j\ ,\ where\ \ j \in NC$
{
IF $CH_j\ =\ MN$ THEN
    $CHj$ transmits a large number of advertisement messages
        $(ADV\_CH)$ having a strong transmission power.
ELSE
    $CHj$  transmits a normal advertisement message $(ADV\_CH)$
ENDIF
$x$ and & $CMs$ will come together with $CH_j$
$CH_j$   establishes a TDMA schedule
$x$ and $CMs$ deliver data to $CH_j$ within the relevant TDMA time frame
}
```

3.4.6. Scheduling attack

During the setup phase of the LEACH protocol, CHs develop TDMA schedules for the data transmission time slots, and scheduling assaults take place. Every node will receive the same time slot for data transfer from the attacker, assuming the identity of CH. Change from broadcast to unicast TDMA scheduling to do this. Packet collisions brought on by these modifications result in data loss. The procedure of scheduling (TDMA) attack is shown in Algorithm 4. The scheduling attack is executed by setting the same time for each cluster member (CM) to send their data packet. Attackers can affect the network in various ways, such as wasting energy on nodes or dropping data packets, which negatively impacts the services of WSNs. Therefore, a technique to recognize these types of assaults and protect the different WSN services is desperately needed.

Algorithm 4 Scheduling (TDMA) attack model

```
$N$ → refers to the size of the Network
$SN$ → refers to the node for sensor
$MN$ → refers to the Malicious Node
$CH$ → refers to the cluster Head
$BS$ → refers to the Base Station
$CM$ → refers to the cluster Member
$NC$ → refers to the list of the cluster heads
$x$ → refers to the value between 0 and $N-1$ (integer value)
$\forall SN_i,\ 0 < i \leqslant N,$ Calculate $T(SN_i)$ and random $r_{SN_i}$
**IF** $(r_{SN_i} < T(SN_i))$ **THEN**
    $SN_i = CH$
**ELSE**
    $SN_i = CM$
**ENDIF**
$\forall CH_j,\ where\ j \in NC$
{
$CH_j$ transmits advertising messaging $(ADV\_CH)$
$x$ and $CMs$ will join $CH_j$
**IF** $CH_j = MN$ **THEN**
    $CHj$ generates a timetable using TDMA to execute the attack and gives the node
        with same time slot for data transmission
**ELSE**
    $CH_j$ creates a typical TDMA timetable
**ENDIF**
$x$ and $CMs$ transfer data to $CH_j$ during the relevant TDMA time slot
$CH_j$ delivers compiled data to BS
}
```

The use of several intelligence and data mining techniques is made possible by the WSN-DS dataset. Consequently, sensor nodes will be more capable of identifying typical behaviors and warning signs of intruders, enabling a prompt to take appropriate action. To evaluate the effectiveness of our proposed model, we compute the accuracy metrics for the used dataset in recognizing and classifying the four distinct types of DoS attacks, based on using improved bi-LSTM in this work [1].

3.4.7. Description of KDDCup99 dataset

DARPA produced the 'KDDCup99' data collection using network traffic obtained from the 1998 dataset [24, 25, 29]. Different 41 features are produced for every network connection through the preprocessing step. The 'KDDCup99' dataset has 4,898,430 records, higher than the total number of records in all other datasets combined. The DoS attack, unauthorized remote access (R2L), unauthorized root access (U2R), and probing (Probe) are the four primary attack categories, as shown in Table 3. In the 'KDDCup99' dataset, many data mining techniques have been applied to detect network traffic breaches.

Table 3. Category of KDDCup99 dataset for attacks.

Attack category	Attack type
Denial-of-Service (DoS)	Worm, udpstorm, processtable, apache2, Pod, smurf, teardrop, back, Neptune, and land (10).
R2L	Examples include FTP write, guessing passwords, imap, multihop, phf, espionage, warezclient, warezmaster, and snmpguess.
Probe	Mscan, saint, nmap, portsweep, ipsweep, and satan (6).
U2R	Loadmodule, sqlattack, ps, xterm, buffer overflow, and rootkit in Perl (7).

The features of the 'KDDCup99' dataset are split into four subjects: Features dependent on time (23–31), content (features 10–22), hosts (features 32–41), and basic features (features 1–9) [30, 31].

Features of KDDCup99 dataset:
Duration: The amount of time spent connected.
Sort of protocol; protocol-type.
Service: A function of the network destination.
Connectivity or lack thereof is the flag.
Bytes sent from source to destination, measured in source-to-destination (Src-bytes).
DST-bytes: The quantity of bytes sent from the source to the destination.
Land: It will be set to 1 if the source and destination port numbers and IP addresses match and to 0 otherwise.
Wrong fragment: The overall incorrect faulty components in a linkage.
Urgent: The number of packets classified as urgent.
Hot: The total amount of "hot" indications indicates the system directory entry.
The total number of times an attempt to log in has failed.
Entered: The login status (1 in the case of a successful login, 0 otherwise).
Num compromised: The overall count of situations that have been breached.
Root shell: The root shell status is 1 when achieved and 0 when not.
Su-attempted: Value is set to 0 unless the "su-root" command is used, in which case it equals 1.
The term "num-root" refers to the total number of operations carried out as a root.
Num-shells: The quantity of shell requests for that particular connection.
Num-file-creations: The total number of actions taken in the process of creating a file.
Num-access-files: The number of control file entries. A file transfer protocol (FTP) session's "num-outbound-cmds" is the total number of commands sent out.
Set is-host-login to 1 if you are logged in as root or admin, and to 0 otherwise.

If a guest signs in, set is-guest-login to 1; otherwise, set it to 0.
Count: The quantity of connections created at the destination to the same host.
Srv-count: The quantity of connections made to the same port.
Serror-rate: The percentage of connections in count (#23) with the flag (#4) activated on any of the connections—s0, s1, s2, or s3.
Srv-serror-rate: The portion of connections with the Active flag (#4) enabled on $s0$, $s1$, $s2$, or $s3$ that add up to the Srv count (#24).
The error rate is defined as the proportion of connections in count (#23) that have the flag (#4) REJ activated. Meanwhile, the 'REJ' is one flag of 'KDDCup99' dataset flags, where the list of the flags that are used in KDD1999 (or 'KDDCup99') dataset are: ('RSTO', 'RSTOS0', 'REJ', 'SF', 'SH', 'OTH', 'S0', 'S1', 'S2', 'S3', 'RSTR') [32].
The srv-error-rate is the proportion of connections in the srv count (#24) that have the active flag (#4) REJ contained in them.
Same-srv-rate (#23): This is the proportion of all connections in the count that went to the same service providers.
Diff-srv-rate is the proportion of all connections in the count that were sent to various services (#23).
Srv-diff-host-rate: The percentage of connections from the srv count (#24) that were directed to distinct destination computers.
DST-host-count: The overall number of connections utilizing the IP address of the target host.
DST-host-same-srv-rate: The proportion of connections that belonged to the same service and were included among the DST hosts (#32). The percentage of connections that use different services of the total number of hosts is known as the DST-host-diff-srv-rate (#32). When determining the DST host count for SRV, the percentage of DST host connections to the same source port that were considered (#33).
DST-host-srv-diff-host-rate: The percentage of connections to various destination computers in the total number of DST host SRVs (#33).
The percentage of connections gathered in the DST host count (#32) with the flag (#4) raised on $s0$, $s1$, $s2$, or $s3$ is known as the DST host error rate.
The percentage of connections ($s0$, $s1$, $s2$, or $s3$) that have a flag (#4) enabled, as compiled in the DST host SRP count (#33), is the error rate for DST host SRP.
For DST hosts, the proportion of connections (#32) with the enabled flag (#4) REJ represents the error rate.
DST-host-srv-error-rate: When combined with the DST host srv count (#32), this represents the percentage of connections that have the enabled flag (#4) REJ.
Label: Attacks' class label.

4. Proposed method

This section provides a detailed explanation of the suggested DL-based model for WSN intrusion detection. The main focus of our proposed model is to use a new improved Bi-LSTM algorithm to solve the intrusion detection problem in WSN systems. The full details of our proposed model are illustrated in Figure 5.

The full details of the proposed model will be discussed as follows. The first section presents a robust deep learning model, an enhanced Bi-LSTM algorithm, for intrusion detection in WSN. In the second section, preprocessing of the raw 'WSN-DS' dataset by converting categorical traffic labels into numerical values, and preparing it for the model. The architecture of the Bi-LSTM model involves the integration of embedding, bidirectional recurrent neural networks, and fully connected layers, which in turn leads to better pattern recognition through capturing temporal dependencies between sequence elements. The accuracy of the model has been observed to increase while its losses keep decreasing within ten epochs when trained on 80% training data from the dataset. During this process, the generalization power and abilities of intrusion detection can be observed within this model.

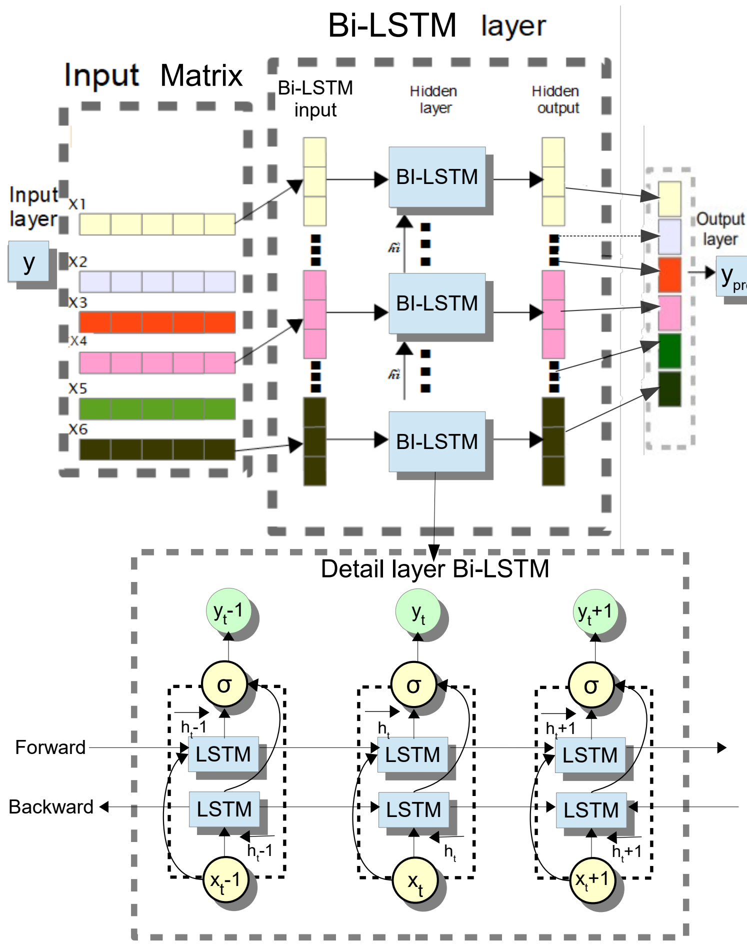

Figure 5. Flowchart of the proposed model.

4.1. Bi-LSTM deep learning model analysis

This section introduces our newly proposed model, a potent deep-learning model for intrusion detection in WSN systems. It is built based on a new enhancement for the Bi-LSTM algorithm. In essence, we modified the earlier LSTM model created in [33, 34]. Then, the Bi-LSTM model, a specific type of recurrent neural network (RNN), is a fantastic choice for figuring out temporal connections in sequences. Because bi-directionality allows the model to use context from both the past and the future while making predictions, it facilitates a richer sequence comprehension.

Input Gate (i_t) in Eq (4.1): The input gate determines the amount of new data that must be stored in the cell. A Sigmoid activation function is used to generate values between 0 and 1:

$$i_t = \sigma(w_{xi}x_t + w_{hi}h_{t-1} + w_{mi}m_{t-1} + b_i). \tag{4.1}$$

Forget Gate (f_t) in Eq (4.2): The forget gate ensures the retention of necessary information from the previous cell state. Like the input gate, it uses a Sigmoid activation function:

$$f_t = \sigma(w_{xf}x_t + w_{hf}h_{t-1} + w_{mf}m_{t-1} + b_f). \tag{4.2}$$

Memory Cell (m_t) in Eq (4.3): The memory cell updates its state based on the forget (f_t), and input (i_t) gates:

$$m_t = f_t \odot m_{t-1} + i_t \odot \tanh(w_{xm}x_t + w_{hm}.h_{t-1} + b_m) \tag{4.3}$$

Output Gate (o_t) in Eq (4.4): The output gate regulates the amount of data passed to the next layer:

$$o_t = \sigma(w_{xo}x_t + w_{ho}h_{t-1} + w_{mo}m_t + b_o). \tag{4.4}$$

Hidden State (h_t) in Eq (4.5): The hidden state of the LSTM cell is computed using the output gate and memory cell:

$$h_t = o_t \odot \tanh(m_t). \tag{4.5}$$

Bi-LSTM Output: The final hidden state in a Bi-LSTM is obtained by concatenating the forward and backward LSTM outputs:

$$h_t^{\text{Bi-LSTM}} = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]. \tag{4.6}$$

Classification Layer: The final output is generated by passing the Bi-LSTM features through a dense layer with Softmax activation:

$$y = \text{softmax}(W_{out}h_t^{\text{Bi-LSTM}} + b_{out}). \tag{4.7}$$

The second-order homogeneous numerical methods proposed in the study of [35] were used to solve differential problems with perturbed turning points. These methods ensure uniform convergence in networks affected by perturbations, which contributes to improving the accuracy of advanced models such as Bi-LSTM algorithms in processing variable data in intrusion detection applications in wireless networks.

4.2. Data preprocessing

We first extracted and transformed the raw ‘WSN-DS’ dataset. Three steps of the preprocessing sequence were executed, making the data acceptable for analysis, emphasizing the use of the Bi-LSTM model. The first step involved turning the spoken terms for the five categories of traffic — Normal, ‘Flooding’, ‘TDMA’, ‘Grayhole’, and ‘Blackhole’ - found in the dataset’s last column into numerical values. This crucial column, which is now referred to as the traffic class column, became the pivotal point for further investigation and model building. These linguistic terms functioned as labels for the respective traffic of various types in every row and are crucial to the classification process of our proposed Bi-LSTM model. The dataset is divided into sets for testing and training. (0.20) of the data will be used as the testing set, and (0.80) will be used as the training set.

4.3. Analysis procedures

4.3.1. Implemented bidirectional LSTM model

Three layers comprise the two-dimensional LSTM model: The dense layer, the two-dimensional LSTM layer that returns sequences equal to true, and the embedding layer. The embedding layer, which is the top layer, converts encoded words into dense vectors. This crucial step captures the semantic relationships between words in the input string. The initial two-dimensional LSTM layer is the second layer. This class processes sequences in both directions and returns sequences for each time step. Because the return sequences = True parameter was used, the layer produced multiple output sequences at the last time step instead of just one. When layering many LSTM layers, this is really helpful. Layer three contains the Bi-LSTM layer two. This layer processes the sequences produced by the previous layer to form a condensed representation. Unlike the previous layer, which returned sequences, it produced a single output for the whole input sequence. The last layer was the dense layer, which has a Softmax activation function. This layer is the output layer of the model for each input sequence. It produces a probability distribution over the classes. The number of units in this layer is decided by the number of classes in your classification task.

There are certain measures taken to ensure the convergence of the Bi-LSTM model. Hyperparameter tuning is the first step, where learning rate and batch size are tuned, as well as the selection of the Adam optimizer for stabilization and speed-up of convergence. Early stopping is also applied, wherein validation loss is monitored so that overfitting can be prevented, and the training process is halted when validation loss stops decreasing. It's important to incorporate certain techniques to ensure smooth convergence and improve generalization. Regularization methods like L2 regularization and dropout can be beneficial. Additionally, to prevent issues such as exploding or vanishing gradients, we can use gradient clipping and appropriate weight initialization methods like He initialization or Xavier initialization. Finally, both training and validation loss curves are monitored to ensure model convergence is carried out properly without oscillation or divergence.

4.3.2. Training model

Using a batch size of 64, our proposed Bi-LSTM model is trained and validated over ten epochs. Throughout the training phase, the model showed a positive learning trend, steadily increasing its accuracy, and decreasing the training loss throughout subsequent epochs. This suggests that the proposed model is successful in identifying dependencies and patterns in the training set. A standard benchmark is used in the validation process for the proposed model. Accuracy measures and loss across epochs improve the final outputs of the proposed model, particularly the functional aspect of the model with unknown data.

It is ideal for training and validation measures to closely match, as this shows how effectively the model generalizes. Large deviations or gaps can indicate problems with overfitting or improper fitting.

5. Results and discussion

5.1. Accuracy curves

Accuracy is a critical measure for evaluating the effectiveness of our proposed model in classification functions. It is defined as the proportion of accurately anticipated occurrences to all instances in the dataset. When dealing with unbalanced datasets, accuracy may not always be the optimum statistic, while being an often-used and reasonable indicator of a model's general soundness. Eq (5.1) is the main mathematical formula for computing the accuracy metric.

$$\text{Accuracy} = \frac{\text{Count of Accurate Predictions}}{\text{Number of Predictions Overall}}. \tag{5.1}$$

The training accuracy curve shows a constant rise, indicating that our proposed model based on enhanced Bi-LSTM is gradually improving its classification accuracy on the training set. This will be fully discussed in Subsection 5.3. Meanwhile, the validation accuracy curve gave information about how well the proposed model was generalized. It showed steady growth over epochs without significant variations in an ideal situation. So, it is better to have a smaller test loss because it shows that the proposed model is correctly predicting the test data. A higher proportion of test accuracy suggests that the model performs well in classifying the test samples. Using this deep learning model, the research achieved 100% test accuracy and 0 test loss.

Other experiments are done to check the running results of the standard LSTM model, and to introduce the benefits of using our proposed Bi-LSTM model. So, we found the following outcomes: i) According to the paper published in [27], which used LSTM on the 'WSN-DS' dataset, and got 96.86% accuracy. ii) Another work published in [36], used the standard LSTM for the 'KDDCup99' dataset, which achieved 93.72% accuracy. Furthermore, we implemented new experiments for reapplying the LSTM as a baseline deep learning model to both datasets used ('WSN-DS' and 'KDDCup99'). The results for accuracy by applying the LSTM model are 65% for the 'WSN-DS' dataset, and 94% for the 'KDDCup99' dataset. These results indicate that applying the LSTM model is not good for inclusion. Consequently, these new experiments confirm the advantages of using our new proposed Bi-LSTM model.

5.2. Attack types

The 'WSN-DS' dataset has four attack types, as shown in Table 4. The objective of the classification task is to categorize cases into distinct attack types (Blackhole, Flooding, Grayhole, Normal, and TDMA). The binary 2D matrix represents a sample of the test data, where the content of Table 4 can be interpreted as follows:

- Rows: In the test data, each row denoted a distinct occurrence or observation. For example, the instance with the identity '246296', which could be a sample ID or index, is matched by the first row.
- Columns: Every column denotes a particular kind of attack. The names of the columns are TDMA, Flooding, Blackhole, Normal, and Grayhole.

The 'KDDCup99' dataset, as shown in Table 5, includes a variety of attack types, each of which is distinguished from the others by particular labels in the 'label' column. The labels "Normal" denote normal network behavior; "DoS" denotes instances of Denial-of-Service attacks; "Probe" denotes instances of Probe attacks. "U2R" denotes attempts at unauthorized elevation to the administrator or superuser privileges, and "R2L" (Remote to Local) denotes attempts at unauthorized access from a remote location. The 'KDDCup99' dataset's descriptions of each assault type correspond with those found in the 'WSN-DS' dataset, offering valuable information about the traits and essence of each kind of intrusion.

Table 4. WSN-DS attack type.

Attack ID	Attack name	Description
1	Normal	Normal network behavior
2	Flooding	Denial-of-Service attack flooding the network
3	TDMA	Time Division Multiple Access attack
4	Grayhole	Grayhole attack disrupting network communication
5	Blackhole	Blackhole attack absorbing network traffic

Table 5. KDDCup99 attack type.

Attack ID	Attack name	Description
1	Normal	Normal network behavior
2	DoS	Denial-of-Service Attack
3	Probe	Probe Attack
4	U2R	User to Root Attack
5	R2L	Remote to Local Attack

5.2.1. Confusion matrix analysis

The main essential tool for evaluating classification models is the confusion matrix (CM), which provides important information about how well the deployment of our proposed model performs against various attack types. In this section, we closely study the CM metrics to give an overview of the model's classification capabilities, as shown in Figure 6 for the 'WSN-DS' dataset, and Figure 7 for the 'KDDCup99' dataset.

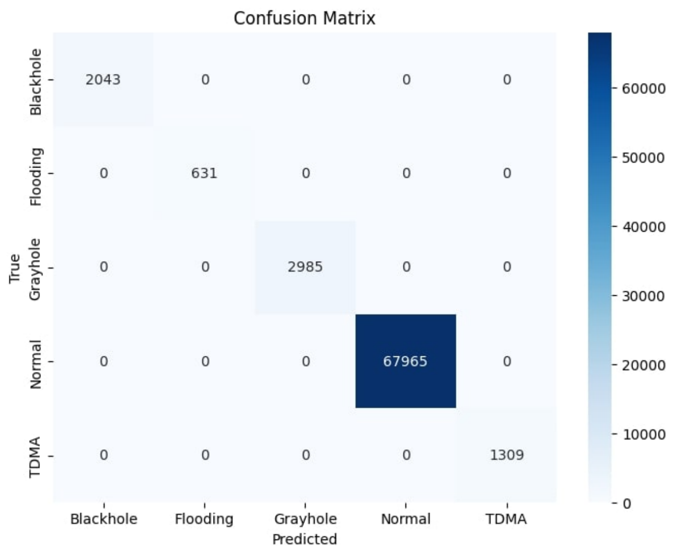

Figure 6. WSN-DS confusion matrix.

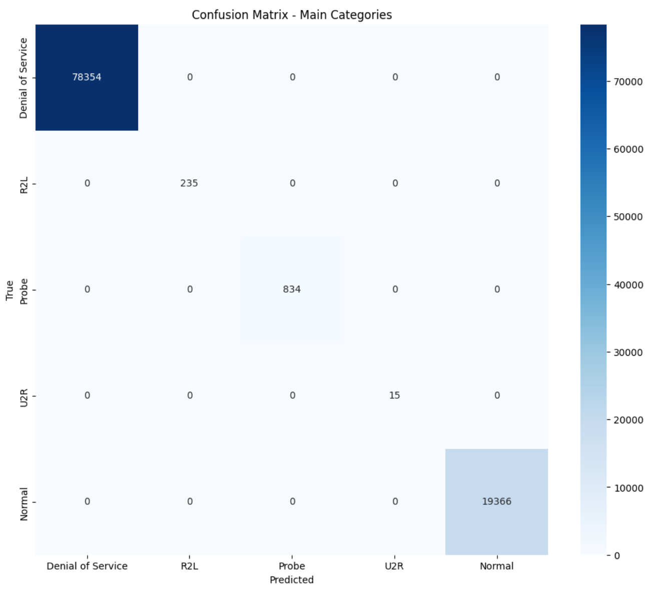

Figure 7. KDDCup99 confusion matrix.

5.2.2. Classification report & discussion

The CM provides a more thorough breakdown of the prediction outcomes as follows:

- True positives (TP): Cases properly classified as the appropriate type of assault.
- True negatives (TN): Cases correctly identified as falling outside the applicable category of assault.
- False positives (FP): These are the cases that are wrongly classified as a kind of appropriate attack.
- False negatives (FN): Incidents that have been incorrectly classified as a distinct type of attack.

In evaluating the accuracy of the proposed model, we can rely on different methodologies, including estimating the ex-ante and ex-post errors mentioned in [37], as these methods are used to ensure the convergence of the final results. This enhances the effectiveness of the proposed Bi-LSTM model in detecting intrusions in wireless sensor networks. To this end, numerical models have been improved by applying advanced methods such as what was proposed in the study [38], which proposed a higher-order numerical method to address quasi-turbulent interaction and diffusion problems with a large time delay.

The following metrics are calculated for each attack type:

- Recall: The percentage of real positive cases is correctly classified by the form of the proposed model.
- Specificity: The percentage of the actual negative cases that the model classifies correctly.
- Precision: The percentage of actual positive cases of all positive cases that have been dropped.
- F1-score: Provides a fair assessment of model effectiveness by taking accuracy and the recall harmonic together [39]. Therefore, F1-Score gives a fair evaluation of the model's efficacy by considering both the recall harmonic mean and accuracy.
- Accuracy: is a measure used to assess the degree of accuracy with which the measurement or prediction of the model was made.

According to the rating report, our proposed model achieved flawless Recall, F1-score, and Precision for each type of attack. The interpretation of the data for the datasets 'KDDCup99' and 'WSN-DS' is as follows:

- Precision (1.00): Each instance that is classified as a specific attack type is already positive. There are no false positives in the model predictions.
- Recall (1.00): Each case of a particular type of attack is appropriately determined; there are no false negatives.
- F1-score (1.00): This score emphasizes a well-rounded performance without sacrificing precision or thoroughness. Furthermore, the harmonic mean of precision and recall is excellent.
- Accuracy (1.00): The model has exceptional overall accuracy, with every prediction matching the true labels exactly.

F1-score, Precision, and Recall values of 1.00 show that achieving a perfect classification performance is a fantastic accomplishment. It suggests that our proposed approach, based on the enhanced Bi-LSTM model, has incredibly well learned the patterns in the data, and it can differentiate between various attack types with zero errors. Taking into account the uneven structure of the dataset, the macro and weighted averages further demonstrate the general quality of the model across all classes. As shown in Table 6 for 'WSN-DS', and Table 7 for 'KDDCup99', to every kind of attack, the performance data for them is also depicted in Figure 8, and in Figure 9.

Table 6. WSN-DS precision-recall analysis.

Attack name	Recall	F1-score	Precision	Support
TDMA	100%	100%	100%	1309
Flooding	100%	100%	100%	641
Grayhole	100%	100%	100%	2935
Blackhole	100%	100%	100%	2043
Normal	100%	100%	100%	67965
Accuracy	100%			74933
Macro Avg	100%	100%	100%	74933
Weighted Avg	100%	100%	100%	74933

Table 7. Kddcup99 Precision-recall analysis.

Class	Recal	F1-score	Precision	Support
Denial of service	100%	100%	100%	78354
Normal	100%	100%	100%	19366
Probe	100%	100%	100%	834
U2R	100%	100%	100%	15
R2L	100%	100%	100%	235
Accuracy	100%			98804
Macro Avg	100%	100%	100%	98804
Weighted Avg	100%	100%	100%	98804

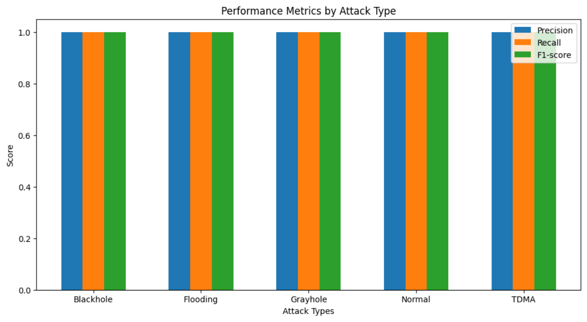

Figure 8. WSN-DS performance matrices by attack type.

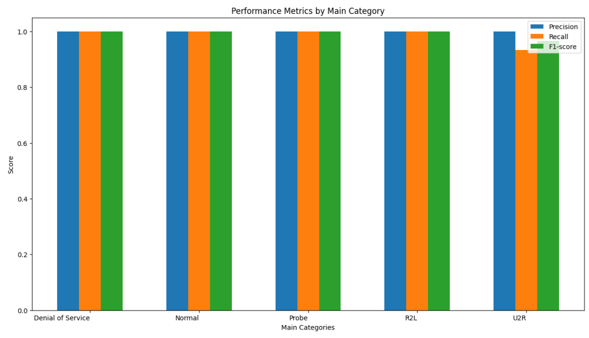

Figure 9. KDDCup99 performance matrices by attack type.

5.2.3. Precision-recall curve analysis

The precision-recall curves show that our proposed model can effectively balance recall and precision in various assault scenarios. Higher numbers indicate better performance of our proposed model. The total performance is evaluated using the area under the precision-recall curve (AUC-PR). A higher AUC-PR indicates better success in striking a balance between recall and precision. Values closer to 1.0, which is achieved by employing our proposed Bi-LSTM model, show a model that successfully captures positive cases while avoiding false positives. An effective method to improve the classification report is the precision-recall curve analysis, which provides more information about how the model behaves at different categorization levels. The precision recall curve is shown in Figure 10 for the WSN-DS dataset, and in Figure 11 for the 'KDDCup99' dataset.

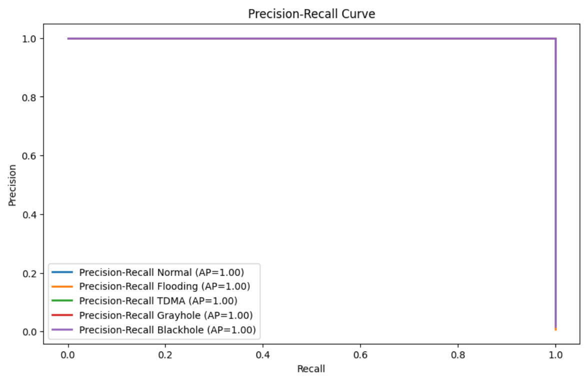

Figure 10. WSN-DS precision-recall curve.

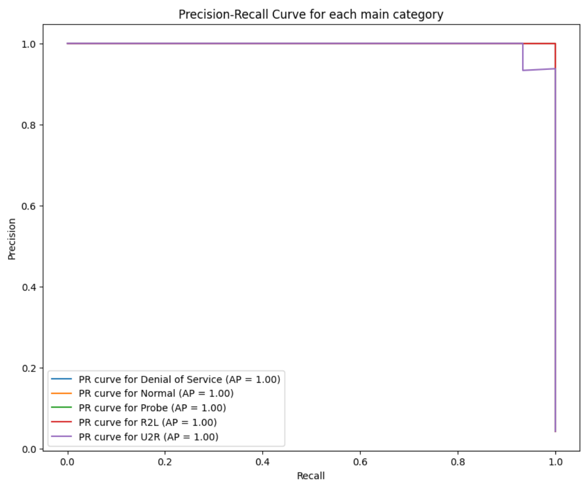

Figure 11. KDDCup99 precision-recall curve.

5.2.4. ROC curve

A graph called a receiver operating characteristic curve, or ROC curve, is used to illustrate the performance of a classification model's overall levels of categorization. The ROC curve represents two parameters displayed, and the rate of TPs and the rate of FPs are also displayed. Since True Positive Rate (TPR) and Recall are interchangeable, Recall can be defined as follows concerning the TPR metric.

$$TPR = \frac{TP}{(TP + FN)}. \tag{5.2}$$

Then, the False Positive Rate (FPR) is defined as follows:

$$FPR = \frac{FP}{(FP + TN)}. \tag{5.3}$$

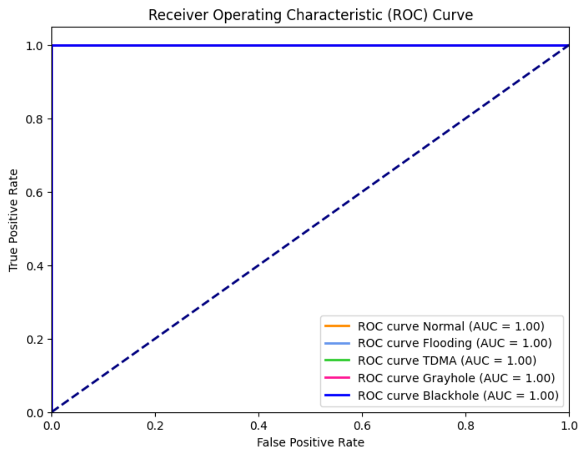

Figure 12. ROC curve for 'WSN-DS' dataset.

The ROC curve is used to plot TPR versus FPR at different categorization criteria. Reducing the threshold for classification results in a higher number of positive classifications, which raises the number of true positives and false positives. One performance indicator used to assess a classification model's capacity for cross-class differentiation is the AUC with the ROC curve (AUC-ROC). AUC-ROC values are expressed as numerical values between 0 and 1. A model with an AUC of 0.5 always outperforms the random walk. A discriminating model is considered perfect when its AUC is 1.0. By computing and analyzing the AUC-ROC values for each attack type, it offers an evaluation of the proposed model's capacity to discern between various kinds of network attacks within the framework of the analysis, as depicted in Figure 12 for the 'WSN-DS' dataset and in Figure 13 for the 'KDDCup99' dataset.

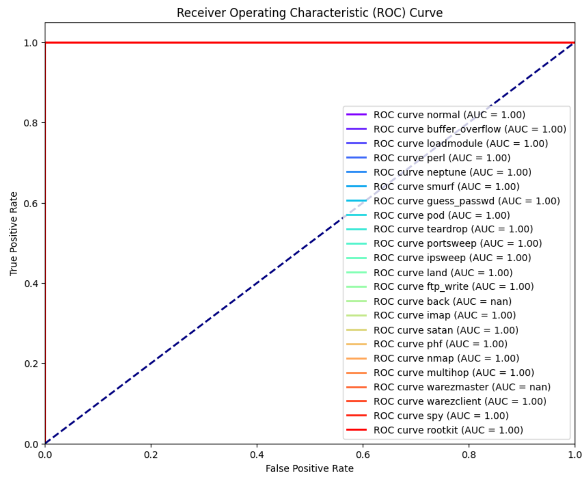

Figure 13. ROC curve for 'KDDCup99' dataset.

5.3. *Learning curve analysis*

The learning curve is a helpful tool for assessing training dynamics and performance over multiple epochs. The following is how the accuracy learning curve is best explained. i) Validation Accuracy: The validation accuracy shows a similar rising trend, indicating that the model can effectively generalize to previously unseen data. ii) Training Accuracy: As the model progresses through epochs, its training accuracy grows steadily, suggesting that it is effectively learning from the training set. The convergence of the training and validation accuracies indicates that the model has been learned and generalized without overfitting. When the training loss gradually decreases, the model reduces errors made during training. Furthermore, the validation loss is trending downward, indicating good generalization ability. If there is a convergence between the training and validation losses, the model might not be overfitting to the training set. Using the WSN-DS dataset in this study, our proposed deep learning model based on Bi-LSTM produced an ideal learning curve, as shown in Figure 14 for the 'WSN-DS' dataset, and Figure 15 for the 'KDDCup99' dataset.

By comparing our proposal based on the Bi-LSTM algorithm with other similar works on the same WSN-DS dataset, as shown in Table 8, the intrusion detection performance reached 100%, the highest score in intrusion classification. To support the proposed algorithm, it is also applied to another dataset, which is the 'KDDCup99' dataset. Compared to similar works, a high classification accuracy is achieved up to 99.9%, as shown in Table 9.

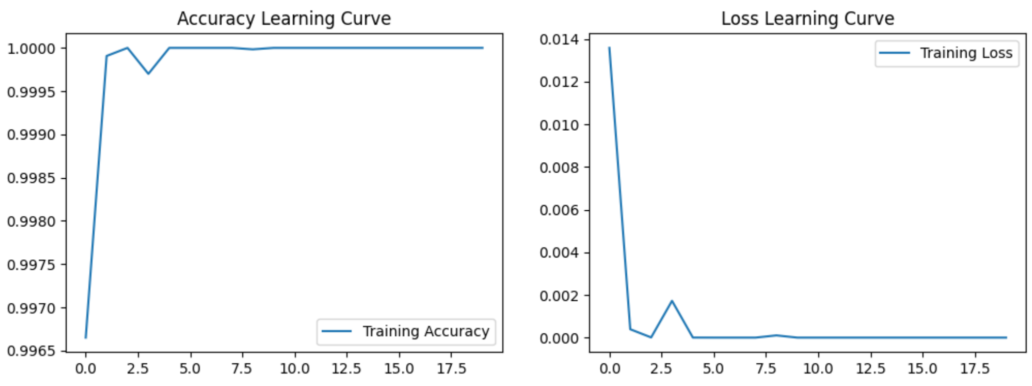

Figure 14. WSN-DS learning curve.

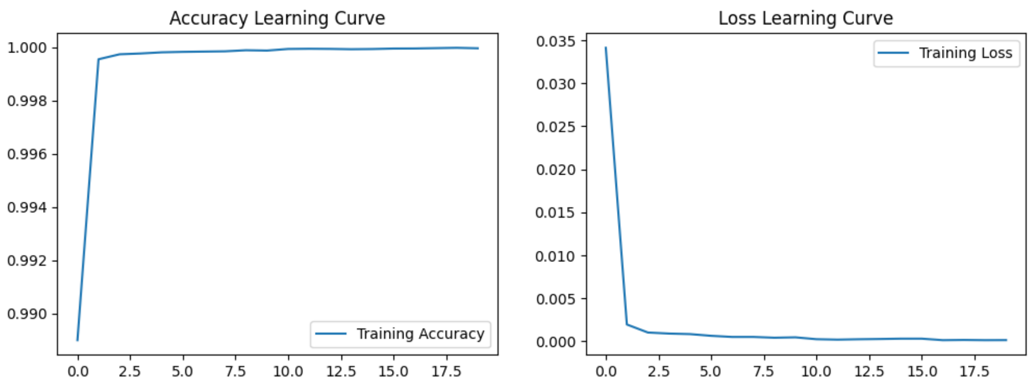

Figure 15. Learning curve for ‘KDDCup99’ dataset.

Table 8. Comparison of Bi-LSTM algorithms and other methods using WSN-DS dataset.

Reference	Method	Used dataset	Accuracy
[10]	Deep learning for WSN intrusion detection.	WSN-DS	The best accuracy is 69.76%
[4]	Machine learning algorithms for WSN security.	WSN-DS datasets.	95.3%
[17]	SLGBM for WSN intrusion detection.	WSN-DS dataset	The best accuracy 99.8%
[1]	WSN-DS dataset for evaluation.	WSN-DS dataset.	The best accuracy is 99.4% for flooding attacks
[27]	CNN, DNN, RNN, and CNN+RNN (LSTM)	WSN-DS	96.86%
Our Proposed model	Bi-LSTM model for evaluation.	WSN-DS dataset.	The best accuracy is **100.0%**

Table 9. Comparison of Bi-LSTM algorithms and other methods using 'KDDCup99' dataset.

Reference	Method	Used dataset	Accuracy
[3]	CSGO-LSVM for enhanced intrusion detection.	NSL-KDD.	0.11
[4]	Machine learning algorithms for WSN security.	KDDCup99 dataset.	92.9%
[6]	Adaptive intrusion detection in WSNs.	Dataset called 'KDDCup99'.	NA
[36]	CNN, LSTM-SA	KDDCup99	93.72%
Our proposed model	Bi-LSTM model for Evaluation.	'KDDCup99' dataset.	The accuracy is **99.9%**

Table 10 presents the hyperparameters used for training the proposed Bi-LSTM model for intrusion detection in wireless sensor networks. It includes the selected values for the learning rate, which controls the speed of weight updates during training, and the Optimizer used to enhance model performance. Also, the BS impacts training stability and learning efficiency. These values were carefully chosen to balance convergence speed and model accuracy. The 'Adam' optimizer was selected to ensure stable weight updates, while BS of 64 was used to optimize training efficiency without excessive memory consumption.

Table 10. Hyperparameter settings.

Hyperparameter	Value
Learning rate	0.001
Batch size	64
Optimizer	Adam optimizer
Dropout rate	0.5
L2 regularization	0.0001
Number of epochs	20
Hidden layer size	[256, 128, 64, 32]
Gradient clipping	5.0
Weight initialization	Default (Keras)

5.4. Assumptions and limitations

This study used some assumptions to obtain the efficacy and feasibility of proposing a new improved Bi-LSTM for intrusion detection in WSNs. The main assumption is that the used datasets are utilized for training and testing, which reflects intrusion scenarios in actual situations, thereby making the model applicable. In addition, all the sensors were presumed to be functioning within normal conditions and without hardware problems or communication delays, which cannot always be so in real deployment. The extraction of features was also presumed to extract adequate patterns for the proposed Bi-LSTM model to identify the intrusions. Some research papers like [40] investigated the complexity performance of the Bi-LSTM model compared to other deep learning models. In addition, it was presumed that the training set contained a large collection of possible intrusion patterns so that our proposed model would be more robust against widely documented attacks.

Consequently, sufficient computational resources were presumed to be available for training and testing to minimize performance bottlenecks. Despite these, this study also possesses several limitations. One of the serious limitations is dataset bias, where the available dataset used does not capture all forms of intrusion types. Another limitation is the computational complexity of Bi-LSTM [40], and its performance in real-time has limited the resource-constrained WSN scenarios. Additionally, environmental factors such as node failures, congestion, or attacks might affect the model's accuracy under real-world deployments. Lastly scalability, since the proposed approach may have difficulty handling very large networks with numerous nodes and dynamic configurations.

6. Conclusions

This study enhanced the security of the communication flow in WSNs and their reliability. The system employs new improvements to the Bi-LSTM model, which is a powerful type of RNN to analyze the input data from a forward and backward perspective. This new improved design of Bi-LSTM helps the model learn sequence dependencies and contextual relationships in network traffic more effectively. The Bi-LSTM architecture has two LSTM networks: one processes data forward and the other processes it backward, allowing for a better understanding of intrusion patterns. Experimental results reveal that our proposed approach effectively detects malicious data transmissions with nearly perfect accuracy, approximately 100% on the 'KDDCup99' and nearly 100% on the 'WSN-DS' dataset. This reflects the effectiveness of using the Bi-LSTM as a strong solution to WSN intrusion detection. Key lessons from this study are the trade-off between the proposed model's performance and computational cost, which can affect real-time use in energy-limited WSNs.

Future research needs to include some optimization methods, like pruning and quantization. Detection must be made robust against different types of attacks, and false positives and negatives must also be addressed. Some models must also learn to be adapted to various environments and enhance security against adversarial attacks, which will present good opportunities for future work and new directions.

Author contributions

All authors contributed equally to this research study, and also to the writing and revising of the paper.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors are grateful for the professional work of the anonymous reviewers, as their valuable comments and suggestions have significantly enhanced the presentation of this research study.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. I. Almomani, B. Al-Kasasbeh, M. Al-Akhras, Wsn-DS: A dataset for intrusion detection systems in wireless sensor networks, *J. Sens.*, **2016** (2016), 4731953. https://doi.org/10.1155/2016/4731953

2. K. M. Nahar, R. M. Al-Khatib, M. Barhoush, A. A. Halin, Mpf-leach: Modified probability function for cluster head election in leach protocol, *Int. J. Comput. Appl. Technol.*, **60** (2019), 267–280. https://doi.org/10.1504/IJCAT.2019.100295

3. D. Hemanand, G. V. Reddy, S. S. Babu, K. R. Balmuri, T. Chitra, S. Gopalakrishnan, An intelligent intrusion detection and classification system using csgo-lsvm model for wireless sensor networks (WSNs), *Int. J. Intell. Syst. Appl. Eng.*, **10** (2022), 285–293.

4. M. S. Alsahli, M. M. Almasri, M. Al-Akhras, A. I. Al-Issa, M. Alawairdhi, Evaluation of machine learning algorithms for intrusion detection system in WSN, *Int. J. Adv. Comput. Sci. Appl.*, **12** (2021).

5. R. M. Al-Khatib, N. E. A. Al-qudah, M. S. Jawarneh, A. Al-Khateeb, A novel improved lemurs optimization algorithm for feature selection problems, *J. King Saud. Univ. Comput. Inf. Sci.*, **35** (2023), 101704. https://doi.org/10.1016/j.jksuci.2023.101704

6. P. Nancy, S. Muthurajkumar, S. Ganapathy, S. Santhosh Kumar, M. Selvi, K. Arputharaj, Intrusion detection using dynamic feature selection and fuzzy temporal decision tree classification for wireless sensor networks, *IET Commun.*, **14** (2020), 888–895. https://doi.org/10.1049/iet-com.2019.0172

7. A. Migdady, A. Al-Aiad, R. M. Al-Khatib, Efficientnet deep learning model for pneumothorax disease detection in chest X-rays images, *Int. J. Bus. Inf. Syst.*, 2022.

8. N. K. Mittal, A survey on wireless sensor network for community intrusion detection systems, in *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, India, (2016), 107–111. https://doi.org/10.1109/RAIT.2016.7507884

9. K. M. Nahar, R. M. Al-Khatib, M. A. Al-Shannaq, M. M. Barhoush, An efficient holy Quran recitation recognizer based on SVM learning model, *Jordanian J. Comput. Inf. Technol.*, **6** (2020), 395–417.

10. H. S. Sharma, A. Sarkar, M. M. Singh, An efficient deep learning-based solution for network intrusion detection in wireless sensor network, *Int. J. Syst. Assur. Eng. Manage.*, **14** (2023), 2423–2446. https://doi.org/10.1007/s13198-023-02090-0

11. M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas, B. Carro, Supervised contrastive learning over prototype-label embeddings for network intrusion detection, *Inf. Fusion*, **79** (2022), 200–228. https://doi.org/10.1016/j.inffus.2021.09.014

12. M. L. Martin, A. Sanchez-Esguevillas, B. Carro, Review of methods to predict connectivity of iot wireless devices, *Novel Appl. Mach. Learn. Network Traffic Anal. Prediction*, (2017), 95.

13. R. M. Al-Khatib, M. A. Al-Betar, M. A. Awadallah, K. M. Nahar, M. M. A. Shquier, A. M. Manasrah, et al., MGA-TSP: Modernised genetic algorithm for the travelling salesman problem, *Int. J. Reasoning-based Intell. Syst.*, **11** (2019), 215–226. https://doi.org/10.1504/IJRIS.2019.102541

14. D. Sarkar, S. Kumar, P. Das, H. Ramos, Higher-order convergence analysis for interior and boundary layers in a semi-linear reaction-diffusion system networked by a k-star graph with non-smooth source terms, *Networks Heterogen. Media*, **19** (2024), 1085–1115. https://doi.org/10.3934/nhm.2024048

15. S. Kumar, P. Das, K. Kumar, Adaptive mesh based efficient approximations for darcy scale precipitation-dissolution models in porous media, *Int. J. Numer. Methods Fluids*, **96** (2024)a, 1415–1444. https://doi.org/10.1002/fld.5294

16. S. Kumar, P. Das, A uniformly convergent analysis for multiple scale parabolic singularly perturbed convection-diffusion coupled systems: Optimal accuracy with less computational time, *Appl. Numer. Math.*, **207** (2025), 534–557. https://doi.org/10.1016/j.apnum.2024.09.020

17. S. Jiang, J. Zhao, X. Xu, Slgbm: An intrusion detection mechanism for wireless sensor networks in smart environments, *IEEE Access*, **8** (2020), 169548–169558. 10.1109/ACCESS.2020.3024219

18. S. Saini, P. Das, S. Kumar, Parameter uniform higher order numerical treatment for singularly perturbed robin type parabolic reaction diffusion multiple scale problems with large delay in time, *Appl. Numer. Math.*, **196** (2024), 1–21. https://doi.org/10.1016/j.apnum.2023.10.003

19. K. Kumar, P. C. Podila, P. Das, H. Ramos, A graded mesh refinement approach for boundary layer originated singularly perturbed time-delayed parabolic convection diffusion problems, *Math. Methods Appl. Sci.*, **44** (2021), 12332–12350. https://doi.org/10.1002/mma.7358

20. M. S. Jawarneh, S. M. Shah, M. M. Aljawarneh, R. M. Al-Khatib, M. G. Al-Bashayreh, Rib bone extraction towards liver isolating in ct scans using active contour segmentation methods, *Int. J. Adv. Comput. Sci. Appl.*, **16** (2025).

21. C. Cai, Y. Tao, T. Zhu, Z. Deng, Short-term load forecasting based on deep learning bidirectional lstm neural network, *Appl. Sci.*, **11** (2021), 8129. https://doi.org/10.3390/app11178129

22. R. M. Al-Khatib, N. K. T. El-Omari, M. A. Al-Betar, Innovative cloud computing object-oriented model to unify heterogeneous data, *Int. J. Oper. Res.*, **46** (2023), 289–322. https://doi.org/10.1504/IJOR.2023.129410

23. N. A. Alrajeh, S. Khan, B. Shams, Intrusion detection systems in wireless sensor networks: A review, *Int. J. Distrib. Sens. Netw.*, **9** (2013), 167575. https://doi.org/10.1155/2013/167575

24. W. Lee, S. J. Stolfo, K. W. Mok, Mining in a data-flow environment: Experience in network intrusion detection, in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (1999), 114–124.

25. W. Lee, S. J. Stolfo, K. W. Mok, A data mining framework for building intrusion detection models, in *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344)*, Oakland, CA, USA, (1999), 120–132. https://doi.org/10.1109/SECPRI.1999.766909

26. M. Dener, C. Okur, S. Al, A. Orman, WSN-BFSF: A new dataset for attacks detection in wireless sensor networks, *IEEE Internet Things J.*, **11** (2023), 2109–2125. https://doi.org/10.1109/JIOT.2023.329220

27. S. Salmi, L. Oughdir, Performance evaluation of deep learning techniques for dos attacks detection in wireless sensor network, *J. Big Data*, **10** (2023), 1–25. https://doi.org/10.1186/s40537-023-00692-w

28. H. S. Sharma, M. M. Singh, A. Sarkar, Machine learning-based dos attack detection techniques in wireless sensor network: A review, in *Proceedings of the International Conference on Cognitive and Intelligent Computing: ICCIC 2021*, Springer Nature Singapore, Singapore, **2** (2023), 583–591.

29. B. J. Santhosh, Kddcup99, 2025. Available from: https://ieee-dataport.org/documents/kddcup99#files.

30. S. Choudhary, N. Kesswani, Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in LoT, *Procedia Comput. Sci.*, **167** (2020), 1561–1573. https://doi.org/10.1016/j.procs.2020.03.367

31. H. Rashaideh, A. Sawaie, M. A. Al-Betar, L. M. Abualigah, M. M. Al-Laham, R. M. Al-Khatib, et al., A grey wolf optimizer for text document clustering, *Int. J. Intell. Syst.*, **29** (2020), 814–830. https://doi.org/10.1515/jisys-2018-0194

32. S. Overflow, Kdd1999 dataset features exolaination, 2025. Available from: https://stackoverflow.com/questions/17024961/kdd1999-dataset-features-exolaination.

33. K. F, Long-short-term memory (lstmwsn_1), 2023. Available from: https://www.kaggle.com/code/kamalfstudify/lstmwsn-1.

34. Q. Abuein, M. Ra'ed, A. Migdady, M. S. Jawarneh, A. Al-Khateeb, Arsa-tweets: A novel arabic sarcasm detection system based on deep learning model, *Heliyon*, **10** (2024), e36892. https://doi.org/10.1016/j.heliyon.2024.e36892

35. P. Das, An a posteriori based convergence analysis for a nonlinear singularly perturbed system of delay differential equations on an adaptive mesh, *Numer. Algorithms*, **81** (2019), 465–487. https://doi.org/10.1007/s11075-018-0557-4

36. K. L. Chiew, B. Hui, An improved network intrusion detection method based on cnn-lstm-sa, *J. Adv. Res. Appl. Sci. Eng. Technol.*, **1** (2025), 225–238. https://doi.org/10.37934/araset.44.1.225238

37. S. Kumar, S. Kumar, P. Das, Second-order a priori and a posteriori error estimations for integral boundary value problems of nonlinear singularly perturbed parameterized form, *Numer. Algorithms*, (2024)b, 1–28. https://doi.org/10.1007/s11075-024-01918-5

38. S. Kumar, P. Das, Impact of mixed boundary conditions and nonsmooth data on layer-originated nonpremixed combustion problems: Higher-order convergence analysis, *Stud. Appl. Math.*, **153** (2024), e12763. https://doi.org/10.1111/sapm.12763

39. I. Abu Doush, M. A. Al-Betar, M. A. Awadallah, A. I. Hammouri, R. M. Al-Khatib, S. ElMustafa, et al., Harmony search algorithm for patient admission scheduling problem, *Int. J. Intell. Syst.*, **29** (2019), 540–553. https://doi.org/10.1515/jisys-2018-0094

40. S. Siami-Namini, N. Tavakoli, A. S. Namin, The performance of lstm and bilstm in forecasting time series, in *2019 IEEE International Conference on Big Data (Big Data)*, (2019), 3285–3292. https://doi.org/10.1109/BigData47090.2019.9005997