JOURNAL OF SOFT COMPUTING AND DATA MINING
e-ISSN: 2716-621X

JSCDM

Vol. 6 No. 1 (2025) 247-261
https://publisher.uthm.edu.my/ojs/index.php/jscdm

Enhancing Intrusion Detection Using Hybrid Long Short-Term Memory and XGBoost

Yousef Alraba'nah[1,2]*, Saleh Al-Sharaeh[2], Ghosoun Al Hindi[2]

[1] *Software Engineering Department, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, JORDAN*
[2] *Computer Science Department, King Abdullah II School for Information Technology, University of Jordan, Amman, JORDAN*

*Corresponding Author: y.alrabanah@ammanu.edu.jo
DOI: https://doi.org/10.30880/jscdm.2025.06.01.016

Article Info

Received: 5 February2025
Accepted: 18 June 2025
Available online: 30 June 2025

Keywords

Intrusion detection, LSTM, XGBoost, network security, RNN

Abstract

The application of Long Short Term Memory (LSTM) networks in Intrusion Detection Systems (IDS) is a promising area of research that leverages the strengths of deep learning in sequence modeling and anomaly detection. This paper introduces a proposed enhancement to IDS by designing a hybrid model combining LSTM networks and eXtreme Gradient Boosting (XGBoost). The paper aim is to highlight the limitations of traditional IDS methods, such as low detection accuracy and high false-positive rates, by leveraging the complementary strengths of deep learning and gradient-boosted decision trees. Moreover, the proposed model is designed for multiclass classification, enabling it to accurately detect and differentiate between various types of network intrusions. The proposed approach improves detection accuracy, reduce false-positive rates, and enhance real-time intrusion detection capabilities, thus providing a robust and efficient solution for network security. The results of the experiments show that the proposed model achieves 98.98% accuracy, 99.03% precision, 99.00% recall and 99.02% f1-score on the testing set. The results approve that the proposed model is outperforming most recently proposed models.

1. Introduction

With the incremental importance of networks in providing vital information, services over the networks are growing at an ever-increasing rate. Which in turn has led to an increase in the number of cyber offense cases. The rise in a number of security threats such as malicious attacks, eavesdropping, and network viruses, has made network security a top priority for both government agencies and society. However, issues can be addressed through intrusion detection. An IDS serves as a mechanism for identifying harmful activities within a system that compromise privacy and security [1]. Typically, detected malicious activities are reported to a network administrator or aggregated in a centrally. Event management and security information systems consolidate data regarding malicious activities from various sources, effectively differentiating between genuine threats and false positives. The initial implementation of an IDS was carried out in 1987, and since that time, research in this field has advanced to offer improved solutions aimed at protecting network systems against a range of network attacks [2].

IDS play a vital role in preserving the network security. However, the rapid growth of online commerce has increased the types of traffic within networks, which results in progressively complex network behavior characteristics, and presents significant challenges to intrusion detection. An inherited issue that must be highlighted is the identification of various malicious network traffic forms, particularly those that are unexpected [3]. IDSs are generally divided into two main categories: signature-based IDSs and anomaly-based IDSs. Signature-based IDSs operate by filtering the signatures associated with known attack patterns. This method is static and is

confined to detecting only previously identified attack patterns. Although signature-based IDSs detect known attacks with very high accuracy and low false-alarm rates, they struggle to detect unknown or novel threats. On the other hand, anomaly-based IDSs use heuristic techniques to detect previously unseen attacks. As the anomaly-based IDSs treat the traffic as an intrusion whenever the deviation from normal traffic patterns is observed, they have a high false positive rate. Nonetheless, in the research field, and as theoretically proven, anomaly-based IDSs, are widely known to detect new forms of attacks [4].

Today's most important models are unable to adequately address the fast and complex nature of networks compromised by cyberattacks. Consequently, the false alarm rate is low and the detection rate and communication and computational costs are prudent [5]. The classical methods of malicious detection include encryption, access control mechanisms, firewalls, etc. But these approaches have built-in limits that preclude full coverage for the network. Notably, intrusion detection is an essential building block for system administrators in a network, to monitor a variety of malicious [6]. Depending on the technique employed for intrusion detection, it is looked as a classifier machine. IDS monitor all network traffic and categorize it as either normal or malicious. This capability allows IDS to employ machine learning techniques to improve classification accuracy. Various methodologies for developing IDS have relied on traditional machine learning methods, including K-nearest neighbors, support vector machines, artificial neural networks, and random forests. These traditional algorithms have several limitations, especially as intrusions are getting more complex and varied [7]. There is a need for improved learning techniques, particularly for automatically identifying and analyzing intrusion features. Since introduced, deep learning has been extensively researched and seen significant success in areas including natural language processing, weather forecasting, and image recognition. Deep learning models have a highly non-linear structure, which allows them to effectively learn from and process complex data [8].

Advancements in parallel computing in recent past has established a robust hardware framework for deep learning methods. Various deep learning techniques are being utilized to build IDSs. This performance gap compared to traditional approaches has been very large as studies show using deep learning approaches achieves significantly better performance than using traditional approaches. Now we have Recurrent Neural Networks (RNNs) which have hit a period of rapid development with the rise of deep learning theory in recent years [9]. However, RNNs have already been widely used in handwritten text recognition and speech processing. Another key feature is that RNNs allow the circulation of information inside a hidden layer. The benefit of this is that the model will remember information from previously processed elements, providing a huge structural advantage in time series data processing. Consequently, many intrusion activities can be represented as specific sequences of events over time within the network. Thus, RNNs are deemed particularly well-suited for the construction of an IDS [10]. To improve the learning capability of IDS and its detection performance, we propose an IDS model based on LSTM and XGBoost. This model combines an LSTM network and an XGBoost classifier in a two-stage process. The LSTM is used to process sequential data to capture important temporal dependencies. The output of the LSTM model is passed to an XGBoost model, which is a powerful gradient-boosted tree model used for classification or regression tasks.

The rest of the paper is structured as follows: Section 2 presents related works. Section 3 introduces the proposed algorithm, detailing its architecture, the dataset used, and preprocessing operations. Section 4 presents the experiments, evaluation metrics, parameter settings, results and comparisons with other works. Section 5 concludes the paper and provides directions for future researches.

2. Related Works

Siviamohan et al. [11] discussed an intrusion detection approach based on RNN Bidirectional Long Short Term Memory (RNN-BiLSTM) that used a two-step methodology in order to overcome the network problem. According to the experimental result, the proposed BiLSTM is better than the other RNN architectures with regards to classification accuracy with a prediction accuracy of 98.48% for the CICIDS2017 dataset. The authors used the random forest and principal component analysis methods in determining the relevant features and filtering out the unwanted feature in the given dataset.

Mushtaq et al. [12] proposed a new intrusion detection framework that uses a two-stage approach by combining Auto-Encoders with Long Short-Term Memory (AE-LSTM) networks. The first stage uses an auto-encoder for dimensionality reduction and feature extraction to capture the important patterns from high-dimensional network traffic data effectively. The second stage uses LSTM networks for classification purposes to model temporal dependencies and sequential anomalies. This increases the detection accuracy to 89.00% with its hybrid approach.

Silivery et al. [13] have designed a deep learning framework that proposes multi-attack classification to improve IDS. Three models are designed in the process: RNN, LSTM-RNN, and Deep Neural Network (DNN). All these models have been trained and validated using benchmark datasets. Benchmark datasets used are as follows-KDD99, NSL-KDD, and UNSW-NB15 for both kinds of classifications. This model was further optimized by using seven different optimizer functions namely Adamax, SGD, Adagrad, Adam, RMSprop, Nadam, and Adadelta, to

determine which one was more effective in this intrusion detection task. It gave the best accuracy with Adamax, at an accuracy of 98.68%.

Kasongo [14] proposed an IDS framework with different forms of RNNs, namely LSTM, GRU, and Simple RNN. The presented framework is tested using benchmark datasets NSL-KDD and UNSW-NB15. To cope with high-dimensional feature space issues, the author performed a feature selection algorithm based on XGBoost. The total number of features was decreased to 17 in UNSW-NB15 and 22 in the case of NSL-KDD, reducing this space to improve detection accuracy and computational efficiency. The paper reports that the proposed LSTM-RNN IDS attained accuracy of 85.93% in network intrusion detection.

Recently, in [15], authors proposed the enhanced LSTM model, called ILSTM, to improve the accuracy and efficiency of IDS. The ILSTM model adapts two swarm intelligence algorithms: Particle Swarm Optimization (PSO) and Chaotic Butterfly Optimization Algorithm (CBOA) - hybrid to optimize weights of LSTM network for good performance with a smaller number of iterations in training. It is divided into two steps, which are: originally, training the weights of the normal LSTM to get the baseline weights, followed by fine-tuning these learned weights using the hybrid optimization techniques from CBOA and PSO. The authors test the ILSTM-based IDS using the NSL-KDD. According to the results, ILSTM achieves an accuracy of 93.09%.

In [16], the authors proposed a hybrid IDS that incorporates both machine learning and deep learning methods. In the hybrid model, LSTM networks act as classifiers while XGBoost and Convolutional Neural Networks (CNNs) feature extraction methods. In this case, benchmark dataset NSL-KDD is preprocessed in selecting relevant features using XGBoost, enhancing the model's ability to identify important patterns associated with intrusions. Further, CNNs are used for extracting more deep features from data, capturing the spatial hierarchies and complex representation. Finally, LSTM networks will process the sequential data to determine the temporal dependency and anomalies signifying potential security breaches. Accuracy of the proposed hybrid model was found to be 98.40%.

The authors in [17] have presented a deep learning framework that integrates RNN with other neural network architectures to enhance the performance of IDS. It combines the capability of sequence learning provided by RNNs with the feature extraction mechanism provided by CNNs. The architecture tries to address major lacunae existing in the conventional intrusion detection systems, namely low detection rates and high false positive rates. The proposed model is evaluated using CSE-CIC-IDS2018 dataset and achieved an accuracy of 98.70%.

3. Methods

In order to address the challenges associated with detection of system intrusion, we propose a comprehensive methodology designed to detect abnormal behavior with high detection rate. The proposed methodology is outlined in figure 1.

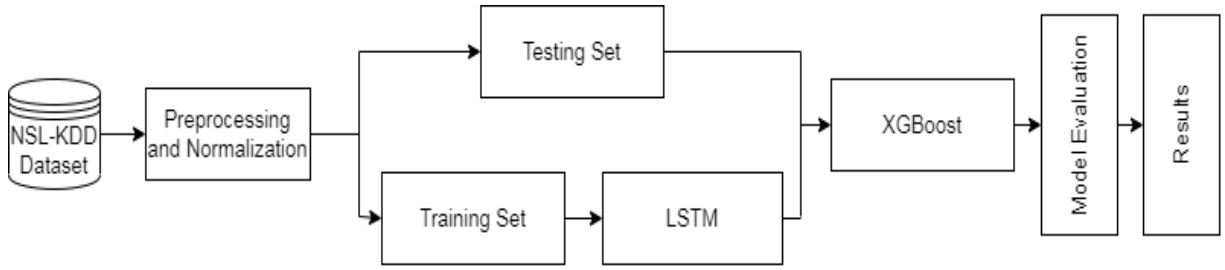

Fig. 1 *The proposed methodology*

The methodology begins with the NSL-KDD dataset, which is a popularly recognized as a benchmark for intrusion detection. Initially, the data is preprocessed and normalized to ensure it is clean, properly formatted, and appropriately scaled for machine learning applications. Furthermore, the dataset is divided into a training set (80%) and a test set (20%). The training set is used for training a model of LSTM, which captures the temporal patterns in the data. Instead of using the final LSTM predictions directly, the hidden state representations are extracted. These learned representations are then used as input for an XGBoost model - a robust gradient boosting algorithm, to enhance the prediction for better accuracy. In model evaluation, the proposed model performance is evaluated performance using the testing set as well as many metrics such as accuracy, precision, recall, and F1-score.

3.1 Dataset Description

Data is a great need for testing the effectiveness and reliability of any intrusion detection model. In fact, the dataset should contain a good amount of records that are valid and reflect real-world network environments very closely.

There have been several IDS datasets that have been generated over these years to meet such requirements. In this paper, one of the most popular IDS datasets is used called NSL-KDD, which is considered the benchmark in intrusion detection. The dataset contains 41 features and 148,517 samples. The NSL-KDD includes normal network traffic as well as four types of intrusions: DoS, U2R, R2L, and Probe [18]. For this particular research, two subsets from the NSL-KDD dataset were used, which are NSL-KDD-Train and NSL-KDDTest+. Table 1 describes all the features in the NSL-KDD dataset. These features are divided into three categorical attributes and the rest as numeric.

Table 1 *NSL-KDD dataset features*

#	Attribute	Type	#	Attribute	Type
1	duration	numeric	22	is_guest_login	numeric
2	protocol_type	categorical	23	count	numeric
3	service	categorical	24	srv_count	numeric
4	flag	categorical	25	serror_rate	numeric
5	src_bytes	numeric	26	srv_serror_rate	numeric
6	dst_bytes	numeric	27	rerror_rate	numeric
7	land	numeric	28	srv_rerror_rate	numeric
8	wrong_fragment	numeric	29	same_srv_rate	numeric
9	urgent	numeric	30	diff_srv_rate	numeric
10	hot	numeric	31	srv_diff_host_rate	numeric
11	num_failed_logins	numeric	32	dst_host_count	numeric
12	logged_in	numeric	33	dst_host_srv_count	numeric
13	num_compromised	numeric	34	dst_host_same_srv_rate	numeric
14	root_shell	numeric	35	dst_host_diff_srv_rate	numeric
15	su_attempted	numeric	36	dst_host_same_src_port_rate	numeric
16	num_root	numeric	37	dst_host_srv_diff_host_rate	numeric
17	num_file_creations	numeric	38	dst_host_serror_rate	numeric
18	num_shells	numeric	39	dst_host_srv_serror_rate	numeric
19	num_access_files	numeric	40	dst_host_rerror_rate	numeric
20	num_outbound_cmds	numeric	41	dst_host_srv_rerror_rate	numeric
21	is_host_login	numeric			

3.2 Preprocessing

The NSL-KDD dataset includes various network attacks, as shown in table 2. These attacks, to reduce the complexity of the dataset, have been grouped into four categories so that detection and analysis models become simpler and effective to focus on a broader pattern rather than an individual signature of the attack. It also helps train machine and deep learning models to recognize general characteristics of attack types rather than specific exploits.

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 247-261

Table 2 *NSL-KDD dataset attacks type*

Attack Category	Attack Type	Count	Total
Normal	No-Attack	77054	77054
Denial of Service (DoS)	apache2	737	53385
	back	1315	
	land	25	
	neptune	45871	
	mailbomb	293	
	pod	242	
	processtable	685	
	smurf	3311	
	teardrop	904	
	udpstorm	2	
Remote to Local (R2L)	ftp_write	11	3882
	guess_passwd	1284	
	httptunnel	133	
	imap	12	
	multihop	25	
	named	17	
	phf	6	
	sendmail	14	
	snmpgetattack	178	
	snmpguess	331	
	spy	2	
	warezclient	890	
	warezmaster	964	
	worm	2	
	xlock	9	
	xsnoop	4	
User to Root (U2R)	buffer_overflow	50	119
	loadmodule	11	
	perl	5	
	ps	15	
	rootkit	23	
	sqlattack	2	
	xterm	13	
Probing (Information Gathering)	ipsweep	3740	14077
	mscan	996	
	nmap	1566	
	portsweep	3088	
	saint	319	
	satan	4368	
Total		**148517**	

StandardScaler is used for standardizing the features of the dataset. The Standard Scaler operates based on the principle of normalization; it scales the distribution of each feature to have an average of zero and a standard deviation of one. This process ensures that all attributes are at the same scale and contribute equally to the model, hence preventing one attribute from dominating the learning process because of its larger scale. Also, the normalizing data has greatly increased convergence speed and efficiency [19]. The formula for standardization is:

$$X_{scaled} \frac{X - \mu}{\sigma} \quad (1)$$

Where X is the feature value to be normalized, μ is the mean of the feature, and σ is the standard deviation.

The preprocessed data further undergos one-hot encoding of features in the data, such as protocol type, service, and flag. One hot encoding is applied to transform raw categorical data into binary vectors. Each category is encoded as a vector, where exactly one of its elements is 1, and the rest are 0. This is the usual procedure for encoding categorical data in cases where categories do not represent an ordinal relationship [20]. Additionally, label binarizer is applied to the class column, thus transforming categorical data into binary format. This is similar to one hot encoding, but has a specific suitability for multi-class and multi-label problems of classification in general [21].

3.3 The Proposed LSTM-XGBoost Model

3.3.1 Recurrent Neural Networks

The RNN is one of the most popular deep-learning architectures it is a type of deep neural network which has become popular because it models sequential data with unprecedented performance. In contrast to multilayer perceptrons or other classical neural networks, RNN is not restricted to operating only in one direction. The feedback of network output to the input makes the RNN efficient for remembering long-term patterns of data [22]. The architecture of a typical RNN is presented in figure 2.

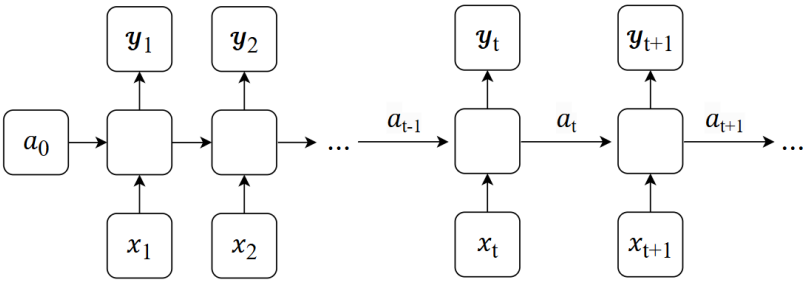

Fig. 2 *The structure of RNN*

A RNN cell takes two inputs at every time step t, namely the current input xt and the previous hidden state at−1. The cell generates two outputs: the current hidden state at and the current output yt. RNNs have become indispensable in processing time series tasks, since they handle data in a sequential manner. Therefore, this would make them highly effective for detecting intrusions unfolding over multiple packets or events [23]. One normal process of training RNNs is done using what was called forward and backward propagation through time, which in definition goes like:

$$a_t = \sigma(W_a \cdot a_{t-1} + W_x \cdot x_t + b_a) \quad (2)$$

$$y_t = \emptyset(W_y . a_t + b_y) \quad (3)$$

The hidden state at time t (at), is computed by the network: Where Wa is a weight matrix associated with the hidden state, Wx is an input weight matrix for the input x and the bias is represented as ba while σ denotes the activation function. In the formula of seconds, Wy is the output weight matrix and by is the output bias, while ∅ is the activation function of the output layer [22]. Even though RNNs are very good at performing various prediction tasks, they still suffer from the problem of a vanishing gradient, a common problem in RNNs. This problem occurs in the training process. This can lead to a slower learning process or even stop the process altogether, whereby patterns cannot be recognized due to contentment with perhaps the first few pieces of information. To avoid this problem, other types of RNNs include LSTM [24].

LSTM is a form of RNN that has taken the lead in sequential data modelling with long-range dependencies. This architecture combines feedforward connections and looping feedback connections, making it possible for LSTM to keep information for a long time. The backbone of this model is based on a unique memory cell in which information can stay for quite a while. Among its salient features, LSTMs use neural network layers known as gates to control information flow across the network. A LSTM cell has three gates: the forget, the input, and the

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 247-261

output gates. The cell unit stores vital information. The forget gate filters out irrelevant data from the previous time step, retaining only the relevant information. The input gate selectively stores relevant information from the current input, effectively filtering out unnecessary data. The output gate specifies which information to output based on the cell state. Gate mechanisms in an LSTM allow for the modification of status values within the cell [25]. The formulas constructed LSTM are presented in the following:

LSTM

$$f_t = \sigma(W_f \cdot [a_{t-1}, x_t] + b_f) \quad \text{forget gate} \quad (4)$$

$$i_t = \sigma(W_i \cdot [a_{t-1}, x_t] + b_i) \quad \text{input gate activation} \quad (5)$$

$$g_t = tanh(W_c \cdot [a_{t-1}, x_t] + b_c) \quad \text{candidate cell state} \quad (6)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot g_t \quad \text{cell state update} \quad (7)$$

$$o_t = \sigma(W_o \cdot [a_{t-1}, x_t] + b_o) \quad \text{output gate activation} \quad (8)$$

$$a_t = o_t \cdot \tanh(C_t) \quad \text{hidden state} \quad (9)$$

The forget gate specifies which parts of the former cell state C_{t-1} should be forgotten. Where W_f is the matrix of the weight for the forget gate, a_{t-1} is the former hidden state, x_t is the current input, b_f is the bias for the forget gate, and σ is the function of sigmoid activation, which gives values between 0 and 1. The input gate manages the adding of new information to the cell state, it composed of input gate activation i_t and candidate cell state g_t. Where g_t holds new information to add, modulated by i_t to regulate how much of this new information is incorporated to the cell state C_t, which is updated by merging the forget gate and the input gate, enabling the LSTM to selectively preserve past information C_{t-1} and integrate new data as required. The output gate determines the next hidden state a_t, which also serves as the LSTM cell output at the current time step o_t [25]. Figure 3 illustrates how LSTM works.

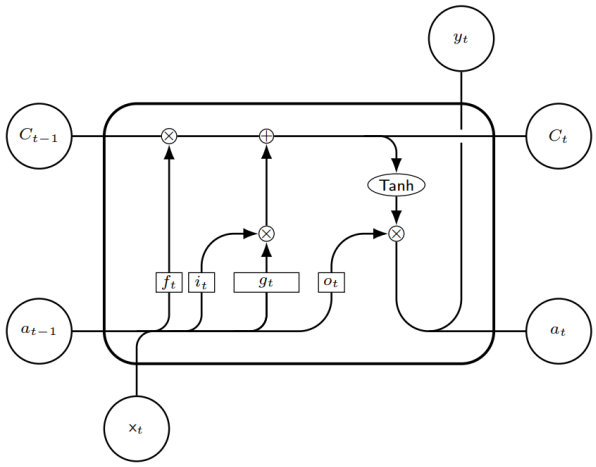

Fig. 3 *LSTM cell work*

LSTM networks are designed to manage and retain long-term dependencies, making them well-suited for sequential data tasks like natural language processing, speech recognition and IDS. In IDS, LSTM can detect intrusions, even those that are unknown or unexpected. There are several advantages of using LSTM compared to other algorithms for IDS. First, the capability of learning long-term dependencies is critical for intrusion detection, spanning over many packets or events. Second, LSTM is more robust against noise, which is very essential in real-world scenarios of IDS systems. Lastly, LSTM is more efficient to train, making it ideal for IDS systems that require training on large datasets [26].

3.3.2 XGBoost

XGBoost is a state-of-the-art high-performance machine learning method that has outstanding performance on structured or tabular input. Therefore, the improvement in predictive performance by XGBoost is largely derived from enhanced computational speed while optimizing for both speed and accuracy. The XGBoost iteratively trains decision trees, adding new trees to fix the errors of past ones, thereby fine-tuning the predictions by eventually combining multiple trees and producing a powerful ensemble model [27]. This is very effective due to its algorithm of a second-order Taylor expansion; hence, it approximates the loss function. Therefore, this is cheap to update

predictions using gradient and Hessian information effectively, hence giving good convergence and excellent accuracy to the model [28].

A key feature of XGBoost is the incorporation of a regularization mechanism that helps to avoid overfitting, penalizing complex models. The regularization term considers the number of leaves in every tree and the weight of the leaves, introducing parameters that adjust the complexity of each tree in order to reduce overfitting. Besides, XGBoost also scales down the contribution of each tree on the learning rate parameter, which makes the model un-sensitive to any of the trees and enhances the generalization capability for it [27]. For a given dataset with n samples, the objective function for XGBoost can be defined as in equation 10, where $L(y_i, \mathcal{y}_i)$ is the loss function, which measures the difference of actual values y_i and predicted values $\mathcal{y}_i$, $\Omega(f_k)$ is a regularization that penalizes each decision tree f_k to prevent overfitting, and k is the number of trees.

$$objective = \sum_{i=1}^{n} L(y_i, \mathcal{y}_i) + \sum_{k=1}^{k} \Omega(f_k) \tag{10}$$

XGBoost, in an iterative process, improves an objective function by adding decision trees that correct the errors of previously set models. Its aim is to minimize a loss function that generally comes through mean squared error for regression and log loss for classification. XGBoost allows for parallelization of the training process, speeding up the processing when big data is used; this is possible because of advanced computational techniques such as cache-aware access patterns and block structure that have been implemented in the algorithm to further optimize memory usage. These discussed innovations make XGBoost a robust option for solving different predictive modeling tasks in many real-world applications with strict latency constraints. This has made the algorithm quite popular for competitive data science and machine learning projects, where it often achieves state-of-the-art performance [29].

Figure 4 illustrates the overall architecture of the proposed LSTM-XGBoost model. The proposed architecture consists of two models: the first employs only an LSTM network for intrusion detection, while the second integrates LSTM with XGBoost to further enhance classification performance. This dual-model approach allows for a comparative evaluation of the benefits gained by combining deep learning with gradient-boosted decision trees. It includes, first, an input layer where the sequential data is presented to the network. Three LSTM layers follow this; these are designed to capture crucial temporal dependencies or patterns in a sequence. Further, deepening the architecture can be achieved by stacking more LSTM layers, thereby gaining insights from complex sequential relationships, particularly on tasks requiring long-term dependency memories. Following the LSTM layer, a flatten layer reshapes the data in a one-dimensional format to suit further dense layers. This is followed by a dense layer, for fully connected processing, in a way that lets the model learn higher-level features from the data. This is then followed by a softmax layer that allows the model to provide probabilities over multiple classes, something very ideal for classification problems. Finally, this model outputs through the output layer, which will present the final predictions of the model. This architecture is well-defined with a flow from LSTM layers to dense and softmax layers, hence targeting high accuracy and performance for sequential data analysis and classification tasks.

The output from the LSTM layers is fed into an XGBoost algorithm for final classification, in conjunction with the ability of the RNN to do sequential modelling. Using the features extracted by the LSTM model, XGBoost develops a sequence of decision trees in such a way that subsequent trees reduce errors from the previously generated trees. Decision trees are built by learning relations among the extracted features with regard to the target variable. This ensemble approach empowers the hybrid model to capture both temporal dependencies through LSTM and complex decision boundaries via XGBoost. The ability of the algorithm to weigh features makes it very suitable for high-dimensional datasets. Further, scalability and support for parallel processing allowed efficient training even on large datasets. This hybrid architecture leverages the strengths of both deep learning and traditional machine learning, and achieves superior accuracy in intrusion detection by marrying temporal analysis with gradient-boosted decision making.

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 247-261

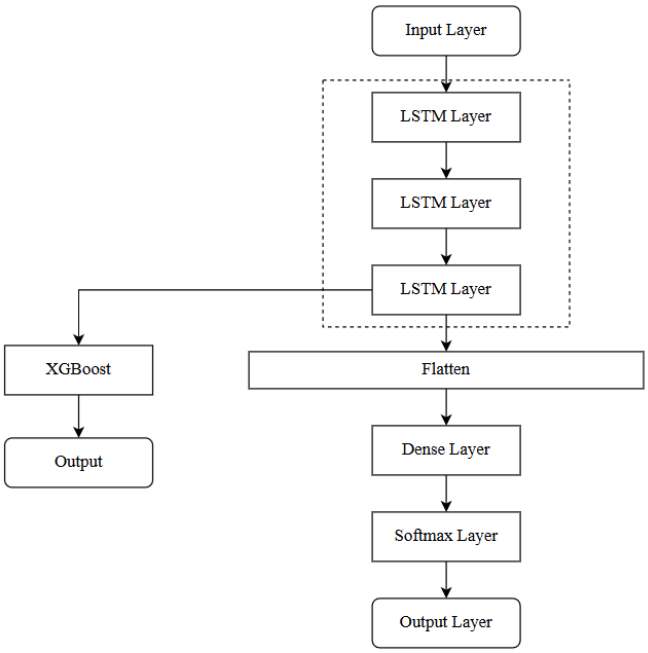

Fig. 4 *The proposed model*

4. Results and Discussion

4.1 Evaluation Metrics

This section shows the evaluation results of the proposed LSTM-XGBoost model on the NSL-KDD dataset. The experiment is conducted using Python with Scikit-learn and Keras libraries. The evaluation is employed by well-established metrics including accuracy, precision, recall, and F1-score to make sure that the model's predictive capability is comprehensively judged. This calculation of metrics considers the values obtained from true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) included in the confusion matrix of Table 3.

Table 3 *Confusion matrix*

	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Where TP means the normal traffic which is successfully predicted as an attack, whereas TN refers to normal traffic that is being predicated successfully as normal. In contrast, FP represents the normal traffic which has been wrongly predicated as an attack while FN refers to the attack predicted as normal.

Accuracy: This gives a holistic view of a model's prediction capability, important in intrusion detection to represent correct identifications of intrusions and normal activities correctly. It is defined as the ratio of the number of correctly classified instances over the total number of instances in any dataset. The metric provides a view on how good the model performs in predicting the TPs, which refer to the intrusion correctly identified, and TNs, which mean correctly identified normal traffic [30].

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{11}$$

Precision: It tells how good a classifier is in identifying TP cases. It calculates the ratio of predicted positive instances that correspond to actual intrusions, reducing false alarms. Precision will play an important role in the evaluation of the classifier performance since it gives an idea of how well it will be able to distinguish real intrusions from benign activities. In security environments, FPs may cause unnecessary alerts, waste of resources, and alert fatigue. With high precision, it gives confidence that if the model detects an intrusion, it will likely be right-a key factor in maintaining efficiency and reliability in the system [31].

$$Precision = \frac{TP}{(TP + FP)} \tag{12}$$

Recall: It tells how good the model is at identifying all the real instances of intrusions, which reduces missed detections. The metric is determined by the ratio of actual intrusion cases that the model identifies to all actual intrusions. Intrusion detection applications usually require high recall to guarantee that threats or unauthorized activities are detected. Failure in intrusion detection might result in undetected breaches, potentially serious security vulnerabilities, and loss of data [32].

$$Recall = \frac{TP}{(TP + FN)} \tag{13}$$

F1-score: It provides the global view in model performance, which is the combination of precision and recall. This is very important within the context of intrusion detection in order to balance the natural trade-off between finding as many intrusions as possible (high recall) and the need for those found to be actually correct-high precision. In fact, the balance is important in security systems: minimizing monitoring actual threats as well as reducing the incidence of false alarms are extremely critical. The F1 score proves to be extremely useful in scenarios characterized by imbalanced datasets, where normal traffic significantly outsmarts intrusions [33].

$$F1 - score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{14}$$

4.2 Model Settings

To assess its effectiveness, the proposed LSTM-XGBoost model is compiled with the Adam optimizer and loss categorical cross-entropy. This architecture will be an excellent fit for the task, since in the multi-class classification problem and allowing the model to learn these complex patterns across multiple output classes from sequential data effectively [34]. In this paper, the Adam optimizer is chosen due to its properties of learning rate that enhance convergence speed and stability, especially in deep neural networks. By combining the Adam optimizer with the categorical cross-entropy loss, we ensure that efficiently the model adjusts its weights and biases to minimize any errors throughout training [35].

This model was trained on the dataset using a total of 100 epochs, with an extra-large batch size of 5000 for memory efficiency and to allow the model to learn from sizable data chunks in each update cycle. With this regard, accuracy and loss on training and validation sets had been checked on the implementation of the 20% split for tracking the performance on unseen data and helping prevent overfitting. During each epoch, the model gets exposed to quite a number of sequences in such a manner that it gets to learn wide temporal dependencies within the data. The experiment trains the model on a substantial number of epochs and a large batch size, while the split is there to make sure that the learned patterns generalize well. This will allow the LSTM layers to see enough data to perform optimally, with a proper balance of hyperparameters to provide an effective and smooth training that yields insight into the model's predictive accuracy and stability across iterations.

After the LSTM model processed the data and learned temporal features, the output (the LSTM features) is fed into an XGBoost classifier. The model's hyperparameters are justified as follows: learning rate is 0.3, number of trees is 100, and max depth is 6. Learning rate is a hyperparameter that specifies the size of the steps the model takes during the gradient descent optimization process. Specifically, it determines how much each tree in the ensemble should contribute to the final prediction. A lower learning rate such as 0.01 or 0.001 means that each tree has a smaller impact, which makes the model learn more slowly and requires more trees (boosting rounds) to converge. This helps improve generalization and can reduce overfitting, but it also increases training time. While A higher learning rate (e.g., 0.5 or 1.0) causes the model to make larger updates, which can speed up the training process. However, it can lead to a model that is less precise and more prone to overfitting. A learning rate of 0.3 is often a good starting point for XGBoost because it strikes a balance between fast convergence and model stability.

The number of trees is the iterations number the model will do while training in the XGBoost model. In each round, a new decision tree is added to the ensemble. Each of these trees tries to correct mistakes from the previously set tree. Usually, adding more trees increases the model's complexity and capability to fit the training data. The cost is that this will increase the overfitting when the trees start to be too complicated in structure, modelling noise in the training data. By applying fewer trees, training accelerates perhaps at the cost of an underfitting model that cannot learn the data relationship with only a few trees. 100 is quite a conventional default and usually has quite a balanced overhead between model complexity and training time. Thus, XGBoost iteratively boosts the model with additional estimators; the model normally can avoid overfitting in further boosting rounds.

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 247-261

The max depth parameter basically specifies the maximum depth of each individual tree in the XGBoost ensemble. That's the way of how complex each tree can get by capping the number of splits or levels a tree can have. A deeper tree has more levels and, hence, is able to model more complex interactions between features. While increasing the depth of trees allows a model to capture more complex interactions present in data, it also increases the risk of overfitting. A deeper tree is likely to memorize the data that it was trained on rather than generalize to new, unseen data. Shallower trees are likely to underfit the data by failing to capture essential relationships and interactions between features. However, they are less prone to overfitting and often generalize better to unseen data. A max depth of 6 is generally effective for many datasets. It allows trees to learn moderately complex relationships between features while still providing some level of regularization to avoid overfitting.

4.3 Results

Figure 5 illustrates the LSTM model training and validation accuracy, which represents the proportion of correctly classified cases out of the total cases. The highest accuracy of the model is 98.92% on the training set and 99.00% on the validation set. The overall average of the accuracy on the training and validation sets are 97.59% and 97.90% respectively.

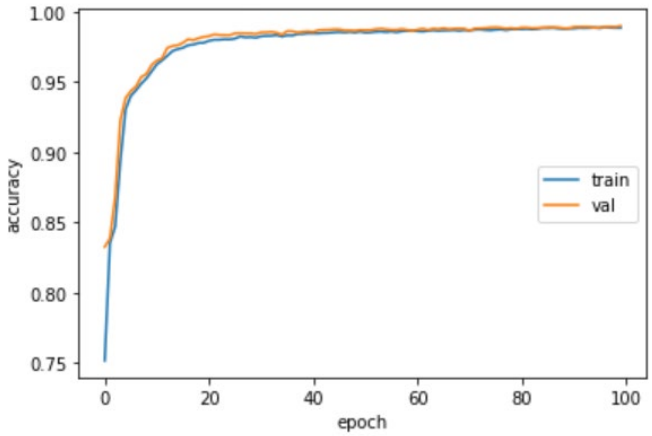

Fig. 5 *LSTM model training and validation accuracy*

Figure 6 presents the evaluation results of the LSTM model across training, validation, and testing phases, highlighting the key performance metrics: accuracy, precision, recall, and F1-score. The LSTM model demonstrates consistently high accuracy, achieving approximately 97.59% and 97.90% during both training and validation, while testing accuracy reaches 98.91%, indicating strong generalization to unseen data. Precision achieves 97.20% and 97.33% during training and validation, with a notable increase to 98.94% in the testing phase. While this is slightly different in recall-the model reaches an approximate 96.07% training and 96.78% validation, and testing recall shows 98.89%-the F1-score shows a balance of precision and recall values close to each other: about 96.63% for the training dataset, around 97.05% for validation, but goes better during testing with 98.91%. The overall outcomes of the validation show how good and robust the LSTM model is. It could be reliably performed during all phases with an exceptionally good performance on the test, hence giving evidence of minimal overfitting and a high predictive capability.

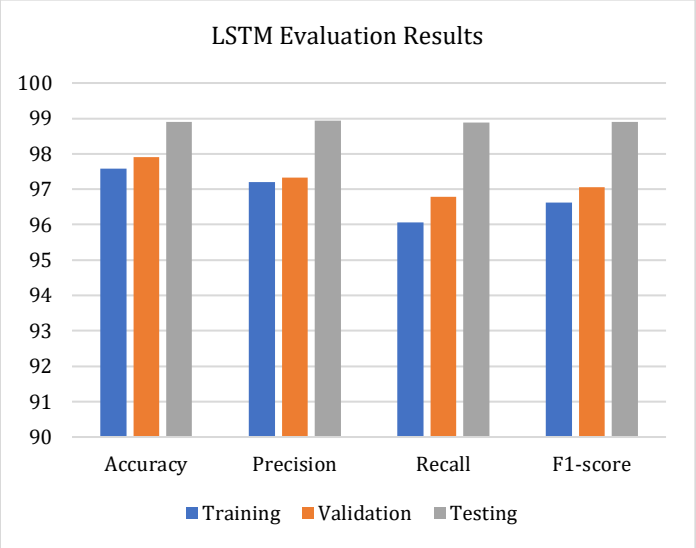

Fig. 6 *LSTM evaluation results*

The performance comparison for both training and testing is presented in Figure 7, showing the results for the evaluation of the proposed LSTM-XGBoost model. It is obvious that the performance is great, touching almost full scores for all metrics. The accuracy, precision, recall, and F1-score have shown 99.53%, 99.46%, 99.65%, and 99.65% respectively during training, while during the testing these values were around 98.98%, 99.03%, 99.00%, and 99.02% respectively. The minimal difference between the training and testing results underlines how well the model generalizes, avoiding overfitting while keeping performance high. The results confirm that the combination of LSTM and XGBoost is a powerful and reliable framework for achieving superior performance in the evaluated task.

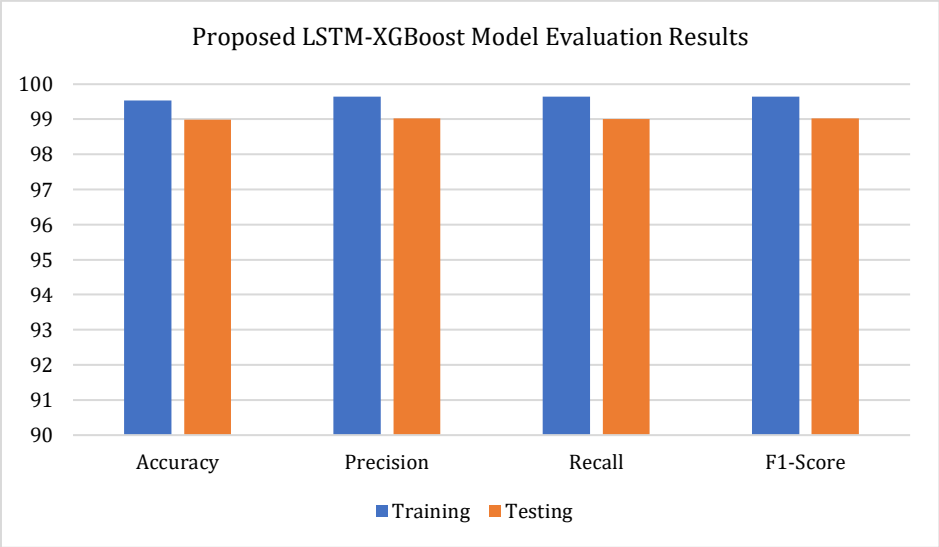

Fig. 7 *LSTM-XGBoost evaluation results*

Figure 8 compares the LSTM model and the proposed LSTM-XGBoost model. From the results of the evaluation, it is quite evident that the proposed model performs better compared to the LSTM in most performance metrics. During the training phase, the proposed model yielded higher accuracy, precision, recall, and F1-score than LSTM, especially in recall and precision. This would therefore show that the proposed model learns better and balances precision and recall well in order to reduce FNs and FPs. During testing, while the gap narrowed, the proposed model still slightly outperformed the other models on all metrics. The proposed model thus consistently outperforms others, reflecting its robustness and generalization capability. Therefore, the model is better suited for real-world applications where the issues of accuracy and reliability are of primary concern.

J. of Soft Computing and Data Mining Vol. 6 No. 1 (2025) p. 247-261

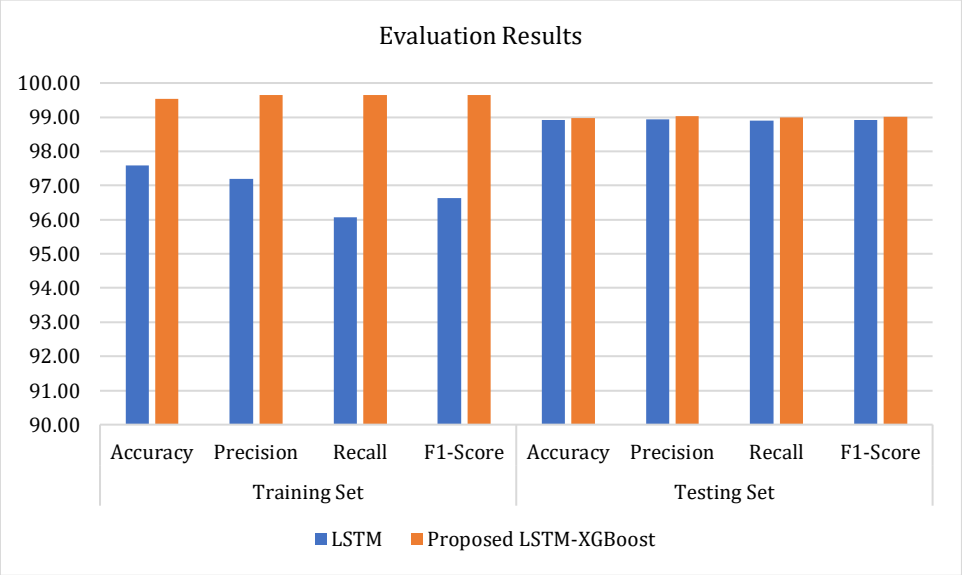

Fig. 8 *LSTM and LSTM-XGBoost comparison*

Table 4 shows a comparison of the most recently proposed deep learning models for IDS. The proposed LSTM-XGBOOST model, demonstrates superior performance with accuracy rates of 98.98%, outperforming all previously reported models in the table. The highest accuracy among prior models is 98.70%, achieved by Rahman's et al. CNN-LSTM [17]. This places the proposed model ahead by a notable margin, highlighting its effectiveness. Additionally, while models like Awad et al. LSTM-RNN (93.09%) [15] and Sajid et al. CNN-LSTM (98.40%) [16] show commendable results, they fall short of the accuracy achieved by the proposed approaches.

Table 4 *Comparison with other models*

Model	Accuracy	Dataset	Method	Publication Year
Siviamohan et al. [11]	98.48%	CICIDS2017	BiLSTM-RNN	2021
Mushtaq et al. [12]	89.00%	NSL-KDD	AE-LSTM	2022
Silivery et al. [13]	98.68%	NSL-KDD	RNN	2023
Kasongo [14]	85.93%	NSL-KDD	LSTM-RNN	2023
Awad et al. [15]	93.09%	NSL-KDD	LSTM-RNN	2023
Sajid et al. [16]	98.40%	NSL-KDD	CNN-LSTM	2024
Rahman et al. [17]	98.70%	CSE-CIC-IDS2018	CNN-LSTM	2024
Proposed LSTM-XGBOOST Model	**98.98%**	**NSL-KDD**	**LSTM-XGBoost**	**2025**

The LSTM-XGBOOST model's integration of sequential learning from LSTM with the ensemble power of XGBOOST provides a distinct advantage, contributing to the improved accuracy. In contrast, most existing models rely solely on LSTM, CNN-LSTM, or RNN architectures, limiting their potential to handle complex intrusion patterns as effectively. Furthermore, the proposed models are benchmarked on the NSL-KDD dataset, ensuring fair and consistent comparisons with the majority of prior works.

5. Conclusion

IDSs play a crucial role in maintaining networks by identifying and responding to potential threats. The proposed hybrid IDS model presented in the paper leveraged the LSTM and XGBoost algorithms. As a matter of fact, traditional IDSs face many challenges with low accuracy and high false positives, which the proposed system has resolved with the help of a hybrid model based on LSTM-XGBoost. This contribution has significantly enhanced the accuracy and efficiency of intrusion detection and has set a base to develop advanced adaptive security systems. The future work will be extensive testing on real network data, investigating other features to increase the performance of the model, and testing with other benchmark datasets.

Acknowledgement

The authors would like to thank Al-Ahliyya Amman University for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

All the authors have designed the research, developed the methodology, performed the analysis, and written the manuscript. All authors have contributed equally.

References

[1] Canpolat, K. M., & Kilincer, I. F. (2024, September). Boosting Based IDS System for Local Network Intrusions. In 2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP) (pp. 1-6). IEEE.

[2] Thakur, K., & Kumar, G. (2021). Nature inspired techniques and applications in intrusion detection systems: Recent progress and updated perspective. Archives of Computational Methods in Engineering, 28(4), 2897-2919.

[3] Neupane, S., Ables, J., Anderson, W., Mittal, S., Rahimi, S., Banicescu, I., & Seale, M. (2022). Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. IEEE Access, 10, 112392-112415.

[4] Abdulganiyu, O. H., Ait Tchakoucht, T., & Saheed, Y. K. (2023). A systematic literature review for network intrusion detection system (IDS). International journal of information security, 22(5), 1125-1162.

[5] Sasikala, G., Laavanya, M., Sathyasri, B., Supraja, C., Mahalakshmi, V., Mole, S. S., ... & Dejene, M. (2022). An innovative sensing machine learning technique to detect credit card frauds in wireless communications. Wireless Communications and Mobile Computing, 2022(1), 2439205.

[6] Heidari, A., & Jabraeil Jamali, M. A. (2023). Internet of Things intrusion detection systems: a comprehensive review and future directions. Cluster Computing, 26(6), 3753-3780.

[7] Zhang, C., Jia, D., Wang, L., Wang, W., Liu, F., & Yang, A. (2022). Comparative research on network intrusion detection methods based on machine learning. Computers & Security, 121, 102861.

[8] Ahmed, S. F., Alam, M. S. B., Hassan, M., Rozbu, M. R., Ishtiak, T., Rafa, N., ... & Gandomi, A. H. (2023). Deep learning modelling techniques: current progress, applications, advantages, and challenges. Artificial Intelligence Review, 56(11), 13521-13617.

[9] Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent neural networks: A comprehensive review of architectures, variants, and applications. Information, 15(9), 517.

[10] Abdulganiyu, O. H., Tchakoucht, T. A., & Saheed, Y. K. (2024). Towards an efficient model for network intrusion detection system (IDS): systematic literature review. Wireless Networks, 30(1), 453-482.

[11] Sivamohan, S., Sridhar, S. S., & Krishnaveni, S. (2021, June). An effective recurrent neural network (RNN) based intrusion detection via bi-directional long short-term memory. In 2021 international conference on intelligent technologies (CONIT) (pp. 1-5). IEEE.

[12] Mushtaq, E., Zameer, A., Umer, M., & Abbasi, A. A. (2022). A two-stage intrusion detection system with auto-encoder and LSTMs. Applied Soft Computing, 121, 108768.

[13] Silivery, A. K., Kovvur, R. M. R., Solleti, R., Kumar, L. S., & Madhu, B. (2023). A model for multi-attack classification to improve intrusion detection performance using deep learning approaches. Measurement: Sensors, 30, 100924.

[14] Kasongo, S. M. (2023). A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. Computer Communications, 199, 113-125.

[15] Awad, A. A., Ali, A. F., & Gaber, T. (2023). An improved long short term memory network for intrusion detection. Plos one, 18(8), e0284795.

[16] Sajid, M., Malik, K. R., Almogren, A., Malik, T. S., Khan, A. H., Tanveer, J., & Rehman, A. U. (2024). Enhancing intrusion detection: a hybrid machine and deep learning approach. Journal of Cloud Computing, 13(1), 123.

[17] Rahman, M. N., & Nijhum, S. R. H. (2024, September). Recurrent Neural Network Based Hybrid Deep Learning Architecture for Enhanced Network Intrusion Detection. In 2024 IEEE International Conference on Power, Electrical, Electronics and Industrial Applications (PEEIACON) (pp. 400-405). IEEE.

[18] Rastogi, S., Shrotriya, A., Singh, M. K., & Potukuchi, R. V. (2022). An analysis of intrusion detection classification using supervised machine learning algorithms on nsl-kdd dataset. Journal of Computing Research and Innovation, 7(1), 124-137.

[19] Imad, M., Abul Hassan, M., Hussain Bangash, S., & Naimullah. (2022). A Comparative Analysis of Intrusion Detection in IoT Network Using Machine Learning. In Big Data Analytics and Computational Intelligence for Cybersecurity (pp. 149-163). Cham: Springer International Publishing.

[20] Ketepalli, G., & Bulla, P. (2023, April). Data Preparation and Pre-processing of Intrusion Detection Datasets using Machine Learning. In 2023 International Conference on Inventive Computation Technologies (ICICT) (pp. 257-262). IEEE.

[21] Kareem, M. K., Aborisade, O. D., Onashoga, S. A., Sutikno, T., & Olayiwola, O. M. (2023). Efficient model for detecting application layer distributed denial of service attacks. Bulletin of Electrical Engineering and Informatics, 12(1), 441-450.

[22] Su, Y., & Kuo, C. C. J. (2022). Recurrent neural networks and their memory behavior: A survey. APSIPA Transactions on Signal and Information Processing, 11(1).

[23] Schutte, C., van der Laan, M., & van der Merwe, B. (2024). Leveraging historic streamflow and weather data with deep learning for enhanced streamflow predictions. Journal of Hydroinformatics, 26(4), 835-852.

[24] Tsantekidis, A., Passalis, N., & Tefas, A. (2022). Recurrent neural networks. In Deep learning for robot perception and cognition (pp. 101-115). Academic Press.

[25] Yadav, H., & Thakkar, A. (2024). NOA-LSTM: An efficient LSTM cell architecture for time series forecasting. Expert Systems with Applications, 238, 122333.

[26] Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent neural networks: A comprehensive review of architectures, variants, and applications. Information, 15(9), 517.

[27] Sarkar, T. (2022). XBNet: An extremely boosted neural network. Intelligent Systems with Applications, 15, 200097.

[28] Zhang, Y., Shi, X., Zhang, S., & Abraham, A. (2022). A xgboost-based lane change prediction on time series data using feature engineering for autopilot vehicles. IEEE Transactions on Intelligent Transportation Systems, 23(10), 19187-19200.

[29] Le, T. T. H., Oktian, Y. E., & Kim, H. (2022). XGBoost for imbalanced multiclass classification-based industrial internet of things intrusion detection systems. Sustainability, 14(14), 8707.

[30] Abualhaj, M. M., Abu-Shareha, A. A., Hiari, M. O., Alrabanah, Y., Al-Zyoud, M., & Alsharaiah, M. A. (2022). A paradigm for DoS attack disclosure using machine learning techniques. International Journal of Advanced Computer Science and Applications, 13(3).

[31] Abualhaj, M. (2024). Enhancing Spam Detection Using Hybrid of Harris Hawks and Firefly Optimization Algorithms. Journal of Soft Computing and Data Mining, 5(2), 161-174.

[32] Nawaz, M. W., Munawar, R., Mehmood, A., Rahman, M. M. U., & Abbasi, Q. H. (2023). Multi-class Network Intrusion Detection with Class Imbalance via LSTM & SMOTE. arXiv preprint arXiv:2310.01850.

[33] Abualhaj, M. M., Hiari, M. O., Alsaaidah, A., Al-Zyoud, M., & Al-Khatib, S. (2024). Spam Feature Selection Using Firefly Metaheuristic Algorithm. Journal of Applied Data Sciences, 5(4), 1692-1700.

[34] Reyad, M., Sarhan, A. M., & Arafa, M. (2023). A modified Adam algorithm for deep neural network optimization. Neural Computing and Applications, 35(23), 17095-17112.

[35] Mahdi, H. F., & Choudhury, T. (2024). LSTM Autoencoders for Internet of Things Data Compression and Battery Conservation. Journal of Soft Computing and Data Mining, 5(2), 151-160.