Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Enhancing intrusion detection: a hybrid machine and deep learning approach

Muhammad Sajid[1], Kaleem Razzaq Malik[1], Ahmad Almogren[2], Tauqeer Safdar Malik[3], Ali Haider Khan[4], Jawad Tanveer[5*] and Ateeq Ur Rehman[6*]

Abstract

The volume of data transferred across communication infrastructures has recently increased due to technological advancements in cloud computing, the Internet of Things (IoT), and automobile networks. The network systems transmit diverse and heterogeneous data in dispersed environments as communication technology develops. The communications using these networks and daily interactions depend on network security systems to provide secure and reliable information. On the other hand, attackers have increased their efforts to render systems on networks susceptible. An efficient intrusion detection system is essential since technological advancements embark on new kinds of attacks and security limitations. This paper implements a hybrid model for Intrusion Detection (ID) with Machine Learning (ML) and Deep Learning (DL) techniques to tackle these limitations. The proposed model makes use of Extreme Gradient Boosting (XGBoost) and convolutional neural networks (CNN) for feature extraction and then combines each of these with long short-term memory networks (LSTM) for classification. Four benchmark datasets CIC IDS 2017, UNSW NB15, NSL KDD, and WSN DS were used to train the model for binary and multi-class classification. With the increase in feature dimensions, current intrusion detection systems have trouble identifying new threats due to low test accuracy scores. To narrow down each dataset's feature space, XGBoost, and CNN feature selection algorithms are used in this work for each separate model. The experimental findings demonstrate a high detection rate and good accuracy with a relatively low False Acceptance Rate (FAR) to prove the usefulness of the proposed hybrid model.

Keywords Intrusion detection system, Machine learning, Deep learning, Cyber security

*Correspondence:
Jawad Tanveer
jawadtanveer@sejong.ac.kr
Ateeq Ur Rehman
202411144@gachon.ac.kr
[1] Department of Computer Science, Air University Islamabad, Islamabad 44230, Pakistan
[2] Department of Computer Science, College of Computer and Information Sciences, King Saud University, Riyadh 11633, Saudi Arabia
[3] Department of Information & Communication Technology, Bahauddin Zakariya University, Multan 60800, Punjab, Pakistan
[4] Department of Software Engineering, Faculty of Computer Science, Lahore Garrison University, Lahore 54000, Punjab, Pakistan
[5] Department of Computer Science and Engineering, Sejong University, Seoul 05006, Korea
[6] School of Computing, Gachon University, Seongnam 13120, Republic of Korea

Introduction

Technologies like cloud computing [1], the Internet, contemporary industrial control systems, automotive networks, etc., have all evolved quickly in recent years. These systems frequently collaborate in symbiosis and manage massive amounts of data using sophisticated communication networks like 5G networks and diverse communication infrastructures [2]. Because of this, a large number of hackers and malicious parties work to develop fresh methods of breaching such computer systems by compromising communication routes. Among the most serious security risks that many organizations face today are network intrusions [3, 4]. Modern Information and Communications Technology (ICT) breakthroughs are incorporated into industrial manufacturing processes by

Springer Open

© The Author(s) 2024. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

the Industrial IoT [5]. The rapid advancement of big data, cloud computing, associated technologies, and information, and our daily communications' increasing reliance on networked services have all contributed to the increased significance of network security [6, 7]. Because of these advancements, networked computing is now essential. The entire network is susceptible to any threat or weakness [8]. Traditional security measures like firewalls and encryption systems are vulnerable to attacks by persistently complex adversaries [9].

Machine learning [10] is used by Network Intrusion Detection Systems (NIDS) and Intrusion Prevention Systems (IPS) to achieve accuracy that exceeds the constraints of existing rule-based techniques based on powerful hardware accelerators and sophisticated machine learning algorithms [11, 12]. Higher computational power hardware accelerators with more processing capacity are becoming available to implement advanced machine learning models [13]. This makes it feasible to accurately identify network breaches and categorize high-capacity traffic inside each session. Attackers are creating unidentified assaults as networks and services grow, leaving the model vulnerable to these attacks [14]. An IDS needs to be smart and efficient at identifying and stopping both known and unidentified threats, like anomaly detection, to protect these networks. The applications of artificial intelligence (AI) to NIDS have become the subject of recent research, and AI-based intrusion detection systems have demonstrated incredible performance. Initially, the primary goal is to integrate well-known machine learning models such as Decision Tree (DT) [15] and Support Vector Machine (SVM) [16] into intrusion detection systems to incorporate deep learning methods like CNNs, LSTMs, and autoencoders. Despite the impressive performance, these results have shown in identifying abnormalities, which also presents issues related to applying them to actual systems [17].

The authors of [18] developed a hybrid intrusion detection model in research for cloud-based systems that can identify all kinds of attacks by combining anomaly and signature-based detection. In another study, the authors of [19] to detect attacks, suggested a novel two-stage deep learning technique that hybridizes long-short-term memory (LSTM) and auto-encoders (AE). The best network parameters for the suggested LSTM-AE are found using the CICIDS2017 and CSE-CICDIS2018 datasets. To boost detection rates while maintaining dependability, the authors of [20] present a novel hybrid model that blends machine learning and deep learning. The suggested approach combines XGBoost for feature selection with SMOTE for data balancing to achieve effective pre-processing. The authors of [21] research provide a method that optimizes the network parameters by combining CNN and GRU for intrusion detection. Various CNN-GRU combination sequences are presented. The CICIDS-2017 benchmark dataset was used by the authors of the simulation, and measures including recall, precision, False Positive Rate (FPR), True Positive Rate (TRP), and other aligned metrics were employed.

In another study, an intelligent and effective Deep Learning network intrusion detection system (NIDS) is presented by the authors of [22]. The authors describe a deep learning-based intrusion detection system (IDS) for attack detection in this work. The CICIDS2018 and Edge IIoT real-time traffic datasets were used to train the model. For Fog nodes and Internet of Things devices to communicate securely and reliably, a high level of security must be maintained. The authors of [23] provide an intrusion detection technique based on artificial neural networks and genetic algorithms to effectively detect different kinds of network invasions on nearby Fog nodes to address this problem. Since various models acquire knowledge about data attributes from disparate viewpoints, the authors of [24] present a hybrid information retrieval system (IDS) in this study that utilizes both random forest (RF) and autoencoder (AE). Two phases make up the hybrid model's operation. Specifically, we use the RF classifier's probability output in the first phase to ascertain whether a sample is part of an attack. The probability output can be utilized to identify unknown attacks. To lower the false positive rate, an extra AE is linked in the second phase. Another [25] study proposes a hybrid intrusion detection model (HIDM) for Industry 4.0 that makes use of transfer learning (TL) and OCNN-LSTM. By applying enhanced CNN parameters obtained by the grey wolf optimizer (GWO) method, the suggested model employs an optimized CNN, which helps to increase the model's prediction accuracy by fine-tuning the CNN parameters. The comparison of the hybrid models with their strengths and limitations is given in Table 1.

Modern benchmark datasets for intrusion detection exhibit class imbalances, with a significantly higher volume of normal traffic than assault traffic despite the wide variety of attacks [26, 27]. This reduces the overall efficacy of NIDS and makes it harder to detect particular types of attacks. Even though inconsistent data negatively affects NIDS's ability to detect assaults, this problem has not gotten enough attention in recent NIDS studies [28, 29]. The current study builds a hybrid intrusion detection classification model based on ML and DL in combination to increase the detection rate (DR) and accuracy. The datasets cover all potential attack methods in the context of Indus experimental IoT and contain rich sample sizes [30, 31]. Network systems are used to transmit diverse and heterogeneous data in dispersed environments. In the meantime, network security, advanced

Table 1 Comparison with Existing Hybrid Studies

Article	Main strengths	Limitations	Algorithm
[18]	Signature-based and Anomaly-based detection	Time Complexity	SDM and ADM
[19]	Identification of the most efficient algorithm	Datasets	LSTM-AE
[20]	Combination of ML & DL,better detection rate	Datasets	RF, CNN, XGBoost
[21]	Feature Optimization, feature learning process	Imbalanced data of attacks	CNN-GRU
[22]	Real time traffic	Longer training time	DCNNBiLSTM
[23]	Distributed architecture of fog computing	Multiclass Classification	Genetic algorithms
Current	Reduced Computational power	Dataset	XGBoost,CNN,LSTM

communication technology, and attack surfaces have grown in the cybercrime era with contemporary digital technologies. Therefore, limiting and possibly even preventing its effects is essential. The core idea of this paper is that creating an intrusion detection system has two main purposes. Initially, the hybrid model looks for unusual activity by tracking network traffic data. It also looks for patterns that change or diverge from typical behavior, as these could be signs of an attack. Second, it notifies personnel in security to look into the situation and take necessary action as soon as an attack is detected. By resolving the following issues, the suggested hybrid XGBoost-LSTM and CNN-LSTM model enhances the current intrusion detection systems:

- It increases generalization and accuracy. Current intrusion detection systems don't detect new types of attacks and don't generalize well. By utilizing the suggested hybrid model XGBoost-LSTM, we can extract feature engineering and manage categorical data rather effectively, which enhances accuracy for a variety of potential attacks. Conversely, CNN-LSTM sequential patterns are recorded in the network to improve generalizability and prevent any unforeseen attacks in the future.

To improve intrusion detection and strengthen network security, the hybrid model that has been suggested has taken care of the following issues:

- The suggested method can handle a wider range of attack detection than the intrusion detection systems that are currently in place.
- The XGBoost algorithm is flexible enough to pick up on fresh information and find previously unnoticed patterns in network traffic.
- CNN can recognize variants of assaults that are not present in training data and can learn sequential patterns from network traffic.
- A high false positive rate is also crucial since relatively few intrusion detection systems now in use produce a lot of false alerts, which is problematic for security staff. We used XGBoost to identify the causes of events, which will aid in reducing the number of false positives.

To address the shortcomings of the current intrusion detection systems, this hybrid approach's primary objectives are to reduce the false positive alert rate, improve accuracy, and generalize to unseen threats.

For this purpose, the main contributions of this research include utilizing machine and deep learning models together to implement a robust intrusion detection system. The main contributions of this paper are given below:

1. We used four IDS benchmark datasets for feature selection using XGBoost and CNN algorithms, and then trained the hybrid model with the help of the LSTM deep learning algorithm using each feature extraction algorithm.
2. We combined the proposed hybrid model with XGBoost-LSTM and CNN-LSTM to train and analyze the performance in terms of several metrics.
3. We demonstrated the practical applicability of the proposed model through the use of test datasets and extensive evaluation with different settings of the hyperparameters.

The remaining structure of this document is described as: We introduced intrusion detection techniques in the introduction section, followed by a brief related work showing how the intrusion detection system was implemented in the previous studies, and then followed by the methodology of the proposed hybrid model with mathematical modeling techniques. In the end, results and a discussion of the hybrid model are presented. The study concluded with a discussion of future directions.

Related work

In recent years, DL and ML methods for anomaly detection have been the subject of numerous studies in the domain of IDS based on AI. The authors of [32] described

an ML-based IDS that combined multivariate correlation analysis (MCA) and LSTM. The information-gain method was the feature selection strategy employed by the MCA-LSTM, in which a subset of features is chosen by the model. The MCA-LSTM achieved 82.15% test accuracy for the 5-way classification using the dataset of NSL KDD, whereas the accuracy of the MCA-LSTM for the 10-way classification job in the UNSW NB15 is 77.74%. Later, the authors in [33] proposed an efficient multi-stage ML-based NIDS framework for NIDS assessment using the RF and KNN algorithms to categorize attacking types. The hyperparameters are optimized using the Tree Parzen Estimator (TRE). The research findings demonstrated that, in comparison to alternative optimization techniques, Bayesian optimization using the Tree-Parzen-Estimator-optimized RF classifier had greater detection accuracy. A hybrid data optimization technique, comprising two components: data sampling and feature selection, is presented. They name it DO_IDS, and it is an effective IDS built on top of this technique. A method for detecting network attacks that integrates deep learning and flow calculations was presented by [34].

Using RNNs, the researchers in [35] developed an IDS based on Deep Learning. The structure of their system contains a data processing block for converting categorical data into numerical inputs, and a scaling function is used to normalize every input, which limits the anomaly detection ability to detect limited attacks. A Feed-Forward Deep Neural Network (FFDNN) is employed in a DL technique for wireless intrusion detection in [36]. The objective was to generate the best possible input subset for the FFDNN classifier to use in identifying network intrusions. For evaluation, the authors took into account the AWID and UNSW NB15 datasets. The AWID dataset is specific to wireless network traffic, unlike the general-purpose UNSW NB15 dataset [37]. A sparse autoencoder-based NIDS is proposed by [38, 39], which stated that the model's multi-classification accuracy on the NSL KDD data set is 79.1%. Similarly, [40, 41] demonstrated that the stacked sparse autoencoder model can be a helpful tool for feature extraction when high-level feature representations of invasive behavior information are extracted.

Some of the researchers have looked into the use of generative models as an additional method of using unsupervised learning to enhance the functionality of current NIDS. They have concentrated mostly on using the fundamental GANs [42], which are based on the Kullback Leibler divergence [43, 44]. After that, in addition to building a variety of GAN models, research has been done to employ appropriate GAN models for particular goals [45]. For this study, we evaluated the effectiveness of the suggested intrusion detection system hybrid model using four datasets: CIC IDS 2017, UNSW NB15, NSL KDD, and WSN DS. By applying XGBoost and CNN, we extracted important features from selected datasets. The extracted feature vector was then used to conduct the training and for evaluation purposes by using experimental procedures. Hence, the presented intrusion detection systems defend against a variety of damaging attacks on systems. In this way, we strengthened the protection of networking devices, which is essential for robust system communication in multi-purpose intrusion detection systems.

Hybrid proposed model

The intrusion detection system gathers and examines security logs, audit data, network behavior, and other network available information. It also makes numerous crucial systemic inferences. It looks for indications that the network or system is under attack as well as whether certain actions are against security policies. Figure 1 displays the general intrusion detection model diagram where it can be seen that the attack has occurred and the intrusion detection system captured the attack and stored it in log files for further actions. The basic intrusion detection model serves as the foundation for the approach put forth in this work. To avoid waiting for the session to end and to reduce the time needed to construct the session feature, it is crucial to employ packet data directly as a feature to achieve real-time detection. The flowchart for the hybrid model is shown in Fig. 2. Finding a packet that can reliably distinguish if an intrusion has happened and detecting the network intrusion based on it are both required at the same time. Current studies are unable to offer this capacity. As a result, this study suggests a novel approach to address this issue.

Data pre-processing

During the data preparation process, the data ranges are changed to improve the compilation and application of the knowledge in a specific dataset. There is a notable contrast shift between the dataset's maximum and lowest range. Data normalization facilitates an approach by reducing the difficulties involved in this process. When applying neural network classification techniques, data normalization has a greater influence. If the neural network learns a backpropagation strategy, input normalization will cause it to speed up training at this point, it will reach maximum efficiency.

Scaling

The normalization of the Americanized data and the differences in the standard deviations and average values of the data read from the CSV file will impact the effectiveness of the learning process. The input data was scaled with Standard Scalar, yielding a standard deviation of one and a mean of zero. Datasets are normalized using library standard scalars by sklearn preprocessing.

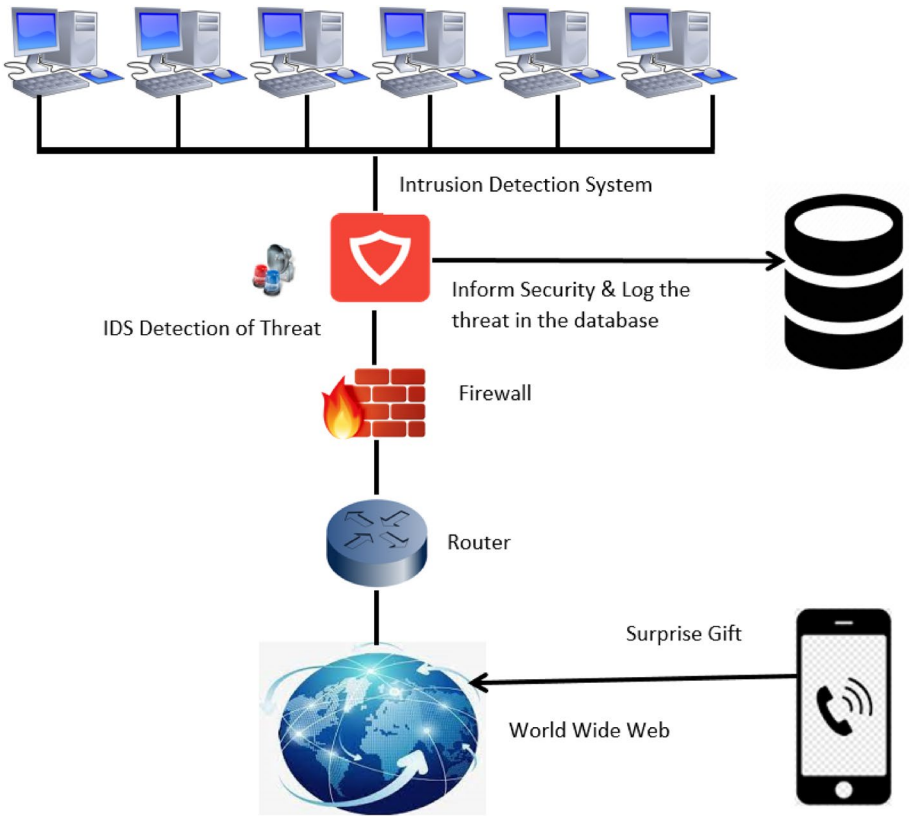

Fig. 1 Intrusion Detection System in General

Regularization technique

L2 regularization is used to determine how comparable the two samples are [46] and the model does not overfit during training. The primary uses of this technique are in text clustering and classification. L2 regularization was chosen because it can highlight certain features with a lesser value but greater significance and weaken the strong features as much as feasible.

Algorithm 1 Algorithm for Min-Max scaling

Require: $[X_1, X_2, ...X_n]$
Ensure: $[X_{tran}[X_1^{tran}, X_2^{tran}, ...X_n^{tran}]]$

while j from 1 to Ins **do**
  if X(j) nominal attribute **then**
Step 1: Apply LabelEncoder
Step 2: Fit and transform the data
    $df \leftarrow X_j$
Step 3: Apply Min-Max scaling

$$X_j^{tran} = \frac{X_j - (X_j)_{min}}{(X_j)_{max} - (X_j)_{min}} \quad (1)$$

Step 4: Repeat Step 3 until data is normalized.
  end if
end while
Step 4: Feed the normalized data to Algorithm 2

Normalization

One preprocessing method for optimizing within-range characteristics is to normalize the data. Data scaling, which uses a minimum and maximum technique to change the net value of the data between [0, 1], is an important component of the normalizing function and the process of normalizing data is shown in Algorithm 1. The following formula gives the normalization formula. The converted input is represented by the expression in Eq. 2. The maximum and minimum values are accordingly represented by the terms $d_{MAX}\, d_{MIN}$. Real value is indicated by the d_i.

$$IN = \frac{(X_i) - (X_i)_{min}}{(X_i)_{max} - (X_i)_{min}} \quad (2)$$

Splitting

75% of the NSL KDD Train is made up of the first subset, which is employed in the training phase. The latter is used in the validation process and makes up 25% of the NSL KDD Train. The UNSW NB15 training set has been divided into two sections, similar to the NSL KDD

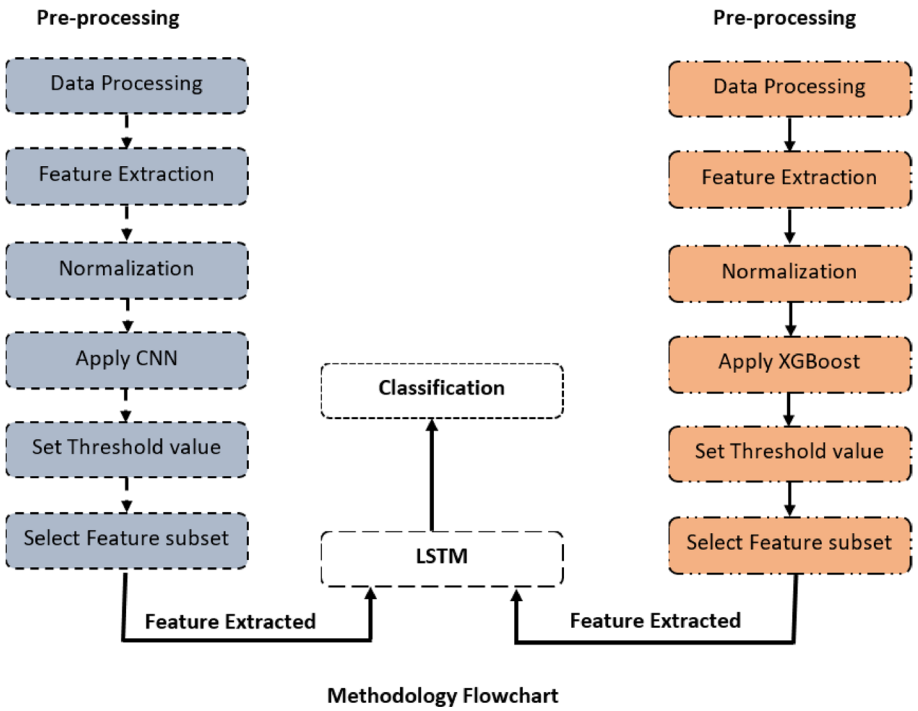

Fig. 2 Flowchart using XGBoost and CNN for feature extraction and LSTM for Classification using extracted features

dataset: the UNSW NB15 Train+ 75% of the original training that is used to train the models, and the UNSW NB15 Val 25% of the original training that is used to validate the trained models. 80% and 20% of the WSN DS and CIC IDS 2017 datasets were split up into training and testing sets, respectively. We separate the training set into validation and training sets to enhance the model's performance even further during training.

CNN and XGBoost are used to extract pertinent features, which enhances the model's performance. Features are chosen by XGBoost based on their relative relevance. To determine the significance of the features, XGBoost takes into account three different kinds of scores. Gain, cover, and weight make up the scoring. Each attribute that has a higher score is given greater weight. Setting a threshold value to choose the features based on scores is another important consideration. When choosing the crucial characteristics, a library called Scikit-learn is utilized to extract features. Relevant features are extracted via convolutional layers. CNN can be used to extract features, which can then be put into an LSTM for classification. The features are chosen using more pertinent, redundant-removal, and prediction contribution criteria in the suggested model. Relevance demonstrates the relative importance of variables that have a high connection with the output variable. Redundancy removal lowers overfitting and enhances model performance. Any attribute that is important for forecasting is chosen. Conventional feature selection techniques are outperformed by XGBoost and CNN since the latter reduces manual labour by automatically learning pertinent features. The features are chosen by XGBoost based on relevance rankings.

Algorithm 2 Algorithm for selection of Optimal variables using XGBoost

Require: normalised input $[X_1, X_2, ...X_n]$
Require: [Y, Output vector]
Ensure: [$X_{Optimized}$, input vector of variables]
Step 1: Declaration of list X_{tran}
Step 2: Declaration of list L_{fk}
Step 3: Initiate XGBoost classifier
Step 4: Input the XGBoost classifier with normalized X and Y
Step 5: Training XGBoost classifier for feature extraction
$X_{tran} \leftarrow importancefactor(FI)$
$FI \leftarrow thresholdFI_{th}$
Step 6:
while item in X_{tran} **do**
  if $FI(item) >= FI_{th}$ **then**
append
    $Lf_i \leftarrow FI(item)$
  end if
end while
Step 7:
Step Sort Lf_i in desc
Generate Optimised X using Lf_i sorted list

Feature selection using XGBoost

The Tree Boosting Algorithm (XGBoost approach) is an ML technique that makes it possible to boost the tree algorithm. XGBoost is comparable to methods like Random Forest (RF) and Extra Tree (ET) algorithms. But unlike Random-Forest and Extra-Tree, XGBoost's trees are not separate from one another; rather, each new Tree enhances the ones that already exist [47–49]. The process of extraction of features using XGBoost is explained in Algorithm 2. Let $Gi = (xi_j, yi_j) * j = 1...p, xi_j \epsilon R_q, yi_j \epsilon R$ represent a given dataset containing p records made up of q attributes indicated by y_i. The result of the model of a group of trees can be written as follows in Eq. 3:

$$\bar{yi}_j = \sigma(xi_j) \sum_{i=1}^{I} fn_i(x_j) \tag{3}$$

where $f_i(x_j)$ is a value assigned to the i_{th} tree of the j_{th} example and fn_i signifies a regression tree. Minimizing the expression in Eq. 4.

$$En\sigma = \sum_{ni} Is(y_i, \bar{y}_j) + \sum_{mi} \phi(fn_i) \tag{4}$$

where the loss connected to loss function En is represented by ls. Additionally, by penalizing Eq. 4 with ϕ, the complexity of the model is decreased. This is how ϕ is defined in Eq. 5:

$$\phi fn_i = \sigma Qt + \frac{1}{2} \upsilon ||\omega||^2 \tag{5}$$

where the length of every weight ω and no. of tree leaves Qt are regularized by σ, υ. To avoid overfitting the model when boosting the tree, ϕ is used. Iterative minimization is the process that takes place. Therefore, fi_t is supplement to the purpose phrase as given at the i_{th} iteration as shown in Eq. 6:

$$En\sigma = \sum_{i=l}^{pt} Is([(y_i, \bar{y}_j))^{ti-1} + fi_t(x_j)] + \phi(fi_t) \tag{6}$$

The Taylor expansion is used in the following ways to simplify the aforementioned phrases (the Loss) in Eq. 7:

$$En_{split} = \frac{1}{2}\left[\frac{(\sum j \epsilon \tau M g_k)^2}{\sum j \epsilon \tau M D_k) + \sigma} + \frac{(\sum j \epsilon \tau L g_k)^2}{\sum j \epsilon \tau L D_k) + \sigma}\right] - \frac{1}{2}\left[\frac{(\sum j \epsilon \tau g_k)^2}{\sum j \epsilon \tau D_k)}\right] - \tau \tag{7}$$

where the collection of instances within the active node is indicated by τ. The node's right side has instances of τM, while its left wing has instances of τL. For our investigation, we generated each feature's FI using XGBoost. We use the XGB Classifier included in the XGBoost Python library [50].

Feature selection using CNN

While LSTM can extract temporal qualities, CNN can extract spatial characteristics. CNN is the first that is utilized because of its capacity to extract high-level attributes from massive amounts of data. The input first passes via the CNN layer and then the convolution layer, where filters are applied to extract the most significant attributes for feature map creation. This map will first go via the Maxpooling layer and then batch normalization to preserve the most noticeable features. Our deep learning model's structure is shown in Fig. 2. Convolution and pooling are the two parts that make up CNN. A series of filters are applied by the convolution layer using a mathematical process. The filter needs to be applied to the input matrix to produce the feature map. The kernel first moves both vertically and horizontally across the input matrix. Until sliding is no longer possible, this process is repeated. This is where the multiplication of the components of the input matrix and the kernel is used to compute the dot product, which is then added up to provide a single scalar output. The feature map is represented by these updated output matrix values. The feature map will be processed by a threshold-based activation function, which will then decide whether or not the neuron fires [51]. An LSTM layer extracts temporal characteristics from the output, a dropout layer is added to stop overfitting. Three iterations of this CNN and LSTM layer combination, each with a different number of hidden layers and kernels, will come before a fully linked layer that performs classification using the SoftMax activation function. ReLU was employed as an activation function in our model in the following Eq. 8.

$$Z_t = hj\left[\sum_{i}^{pt*qt} \omega_i \tau_i + b\right] \tag{8}$$

When τ is the input data, ω stands for weights, h_j for the activation function, b for bias, and pt and qt for the sizes of the input data matrix. Without changing the weights, the Maxpooling approach will decrease the sample size [52]

Batch normalization

Batch normalization (BN) is mostly employed in deep neural networks to avoid covariance changes that happen when input is transferred between layers. These changes result in less efficient learning and instability in the learning process. Batch normalization will decrease

generalization mistakes and quicken the optimization process [8]. Furthermore, it will modify the CNN output via LSTM layer processing after scaling the input layer data to a unit norm. The following formulas contain the batch normalization mathematical representations. ϵ is used to guarantee that the denominator in the formula is non-zero, X_t stands for the data produced by the Max-pooling layer, and μBN and δBN are the batch mean and variance, respectively and are given in the following Eq. 9.

$$\bar{Xt} = \frac{Xt - \mu BN}{\sqrt{(\delta_{BN})^2 + \epsilon}} \tag{9}$$

Two variables, γ and β, will be used to process the output of Eq. 9. By training the min during the learning process, this procedure will produce an output called $\bar{y}$, where γ and β are employed to improve the learning output.

$$\bar{Yt} = \gamma[\bar{Xt}] + \beta \tag{10}$$

Long short-term memory

CNN cannot extract and analyze the associations between sequences and instead focuses on analyzing the internal characteristics of a data packet. Consequently, the model built using the LSTM network will perform better when it comes to training for network intrusion detection. To tackle gradient disappearance caused by recurrent neural networks, LSTM is gradually applied to various network intrusion detection models. The LSTM unit takes the role of the neurons in the recurrent neural network (RNN), which is based on the conventional cyclic neural network. The following are the basic building blocks for an LSTM architecture.

Forget gate

A value of 0 indicates that the information in this bit has been forgotten, while a value of 1 indicates that it has been fully retained. Equation 11 displays the computation formula.

$$G_t = \sigma[\omega_g * (hn_{t-1}, x_t) + b_G] \tag{11}$$

Input gate

To the greatest extent feasible, the input-gate augments the data that the new cell state requires. The current cell state is multiplied by the sigmoid function, which has a value range of $0 - 1$, as the output of the input gate. The following is the calculation formula in Eqs. 12 and 13

$$I_t = \sigma[\omega_{it} * (hn_{t-1}, x_t) + b_I] \tag{12}$$

$$\bar{C_t} = \sigma[\tanh_c * (hn_{t-1}, x_t) + b_C] \tag{13}$$

The final new cell state can then be created by combining the data from the previous and new states. as given in Eq. 14.

$$C_t = [G_t * C_{t-1}] + [I_t * \bar{C_t}] \tag{14}$$

Output gate

The sigmoid function that the output gate produces has a value between 0 and 1. The Eqs. 15 and 16 show how the activation function is computed.

$$OP_t = \sigma[\omega_{OP} * (hn_{t-1}, x_t) + b_{OP}]h_t = OP_t * \tanh C_t \tag{15}$$

$$hp_t = OP_t * \tanh C_t \tag{16}$$

Hybrid model architecture

The proposed hybrid intrusion detection system uses LSTM as the classifier and XGBoost and CNN as feature extraction methods. First, benchmark datasets are fed into the CNN and XGBoost models, each of which then undergoes data preprocessing operations before features are chosen to feed into the LSTM classification to confirm whether or not an intrusion has occurred. Following the model's construction and assessment, the decision classification process is then completed. Figure 3 depicts the interior layout of the LSTM network layer architecture.

First phase of the hybrid model

The core elements of the suggested hybrid model are the CNN and LSTM networks. We chose the CIC-IDS 2017 and WSN-DS datasets for the first phase implementation using the CNN-LSTM network. The two-layer CNN network receives the preprocessed data first, uses feature selection on the traffic data, and selects the global average pooling layer to be the fully connected layer. Data feature extraction and dimensionality reduction are accomplished through the use of convolution and pooling algorithms, yielding a feature matrix as the end product. Next, feed the feature vector into the unidirectional double-layer LSTM network. This network learns and classifies the features selected by the CNN network by utilizing the powerful time series learning capability of the LSTM. The input, output, and storage gates of the LSTM network continuously and repeatedly train using a large amount of data to change their parameters. Next, by using the data from the CNN network to determine the time-fitting relationship between the data, the effective dynamic modeling of the input and output data of the prediction time series is finished. In the end, the CNN-LSTM model matches the learned data and generates the predicted value using a fully connected neural network. Figure 4 illustrates

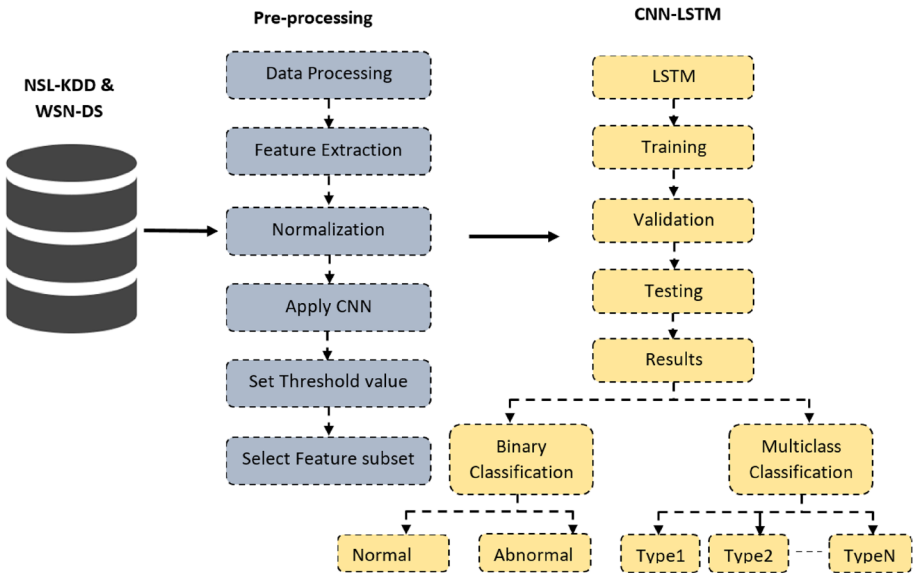

Fig. 3 Framework for IDS using CNN and LSTM

the CNN-LSTM network's basic architecture using the CNN-LSTM model trained in a multi-classification scenario. Below is a detailed explanation of the CNN-LSTM hybrid model:

- Input Layer: This is the model's initial architectural layer. Its function is to take in input data and send it to the layer above. The input data for this architecture is given as a grayscale image with dimensions of 6 by 6.
- Convolutional Layer: By applying specific filters to this layer, convolutional layers are utilized to extract features. This layer applies a 3 x 3 kernel size to every convolutional layer. After applying the kernel, each layer creates a feature map, which is then supplied to the layer that is connected to it next. Three convolutional layers at each layer with filter sizes of 32, 64, and 128 are used in this architecture to extract a variety of characteristics.
- Pooling Layer: Maxpooling is used in this layer to minimize dimensions. To this layer, we applied a 2 x 2 filter that extracts the greatest value from the supplied input. By using this layer, the network's computational costs are decreased, giving this layer more resilience.
- LSTM Layer: The CNN's extracted features are passed into this layer. The ability of LSTM to identify long-term dependencies in sequences is a key characteristic. To record the future values and dependencies of network packets with timer series packets entering the network, we applied the LSTM layer at the end.
- Fully-Connected Layers: These layers take the output from the convolutional layers and flatter it into the shape of a vector. This vector is fed into several fully connected layers to perform a linear combination of the input to produce the final output in the form of classification. The ultimate output layer provides the likelihood of an input falling into a specific class.

The architecture is explained in more detail in the paragraph that follows. After the preprocessed data is flattened into a 1-dimensional array, the data in Fig. 4 is transformed from low-dimensional to high-dimensional using two layers of $3x3$ convolution operations. These channels yield feature maps with 64 and 128 pixels, respectively, and have a height × width of $6x6$. This suggests that the significance of the data represented by various dimensions varies. The only thing impacted, not the feature map's dimensions, is its size, as seen by the fact that the feature map's height and breadth increase to $3x3$ while the pooling kernel shrinks to $2x2$ after the maximum pooling operation. There are still the same amount of channels. Finally, the LSTM model uses the significant features that CNN extracted as input data. The second fully connected layer's task is to carry out output classification with 64 input nodes in the fully connected layer. The total number of classifications needed determines how many output nodes are required. This model has six output nodes because it has six classes. Ultimately, the output probability is transformed into a range between 0 and 1 and the output results are normalized using the Softmax algorithm.

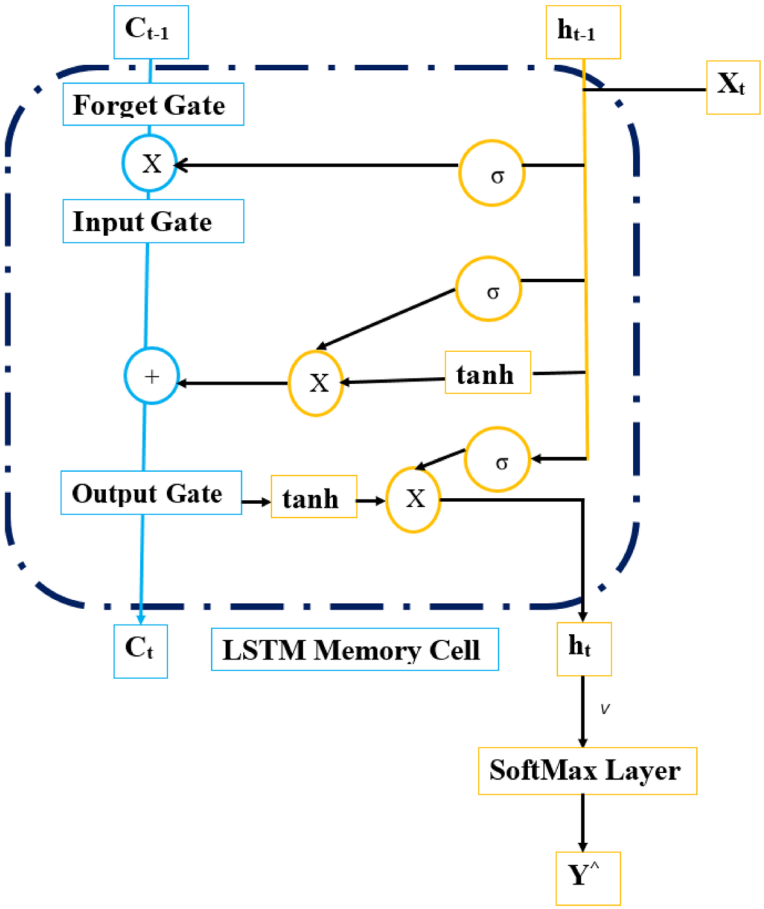

Fig. 4 LSTM Architecture

In this study, Adam is the optimizer, the batch size is 1024, the epoch is 50, and the learning rate is 0.001.

Second phase of the hybrid model

In the second phase, we proposed the XGBoost-LSTM model for binary and multi-classification. The IDS hybrid model using XGBoost-LSTM is shown in Fig. 5. Gathering the information required to develop the models is the first step. Two datasets were taken into consideration in this phase: the UNSW NB15 and the NSL KDD. The Data Processing and Feature Extraction layer is the second layer of the hybrid model. To make sure that every numerical feature has been normalized and that all of the categorical variables are accurately encoded, a specific dataset is normalized in this stage. Following the dataset has been standardized and cleansed, the XGBoost algorithm is used for feature extraction. A vector with the Feature Importance (FI) value is produced by this method, and an FI threshold that has been statistically calculated is used to select the ideal feature subset. The Model Building phase is the third tier of the architecture. Three main operations-training, validation, and testing occur independently during this phase. The performance criteria are used to evaluate each step in this process. In addition, Fig. 3 shows how LSTM networks under investigation are configured. LSTM deep layer design is fed by the input layer, which is the first layer. A Dense NN layer is then used to compute the data from the deep levels. Ultimately, the predictions are calculated using the ReLU in Eq. 17 and the Sigmoid function for the binary classification setting in Eq. 18, or a Softmax in Eq. 19 for the multi-class classification configurations [53]. An activation function with values (probabilities) that sum to 1 is returned by the Softmax [54]. The prediction is represented by the greatest value.

$$ReLU(zt) = max(0, zt) \tag{17}$$

$$Sigmoid(zt) = \frac{1}{1 + e^{-zt}} \tag{18}$$

$$SoftMax(zt)_j = \frac{e^z t_j}{\sum_{k=1}^{M} e^z t_k} \tag{19}$$

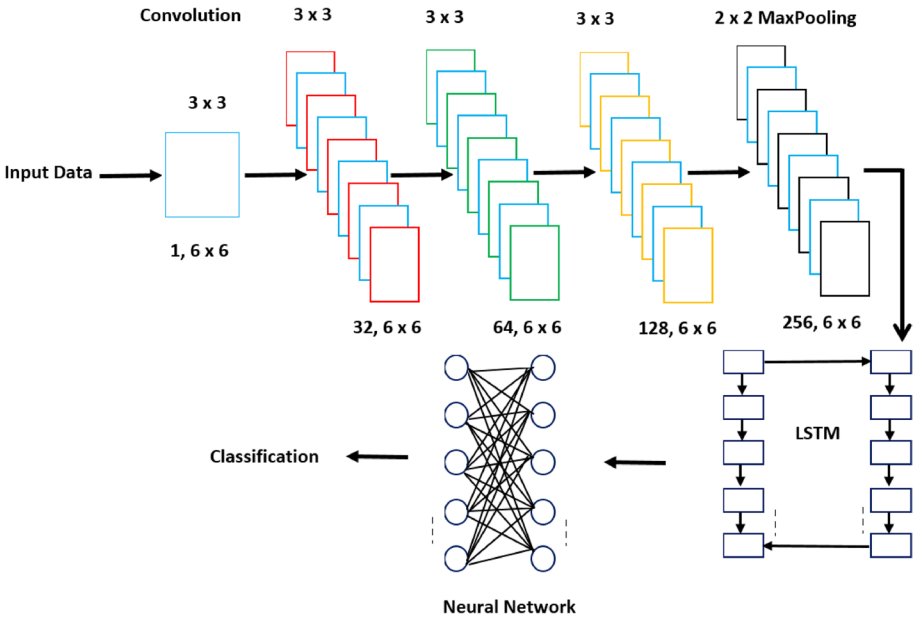

Fig. 5 Hybrid Model Architecture using CNN and LSTM

Results and discussion

Experimental setup

The experiment's hyperparameter parameters and training settings in this study are given in the Table 2: The hybrid model contains additional hyperparameters that can affect how well the model performs.

- Throughout the XGBoost feature extraction process, the max_depth hyperparameter aids in managing the intricate structure of each tree. The ideal balance is found by fine tweaking to prevent the model from being overfit or underfit.

Table 2 Experimental Setup and Hyperparameters

Parameter	Value
Batch Size	256
Learning Rate	10^{-3}
Learning Rate	Adam
Number of Epochs	70
Activation functions	ReLU and SoftMax
Binary Classification Loss	binary_crossentropy
Multiclass Classification Loss	categorical_crossentropy
Processor	Intel Xeon W1370
Windows Platform	10 with 16 GB RAM, SSD
GPU	RTX3070
Language & platform	Python & Jupyter Notebook
Libraries	TensorFlow, Pandas, and Keras libraries.
Classification Methods	binary and multiclass.

- The number of trees in the ensemble is displayed by the n_estimators hyperparameter. While the model's complexity improves with more trees, training time also increases. We adjusted many n_estimators to achieve the ideal combination of efficiency and accuracy.
- Gamma is an additional hyperparameter that is crucial. It separates the more significant nodes. It was possible to avoid overfitting by fine-tuning with a limited number of splits.
- To capture complex temporal dependencies, LSTM layers are stacked. We experimented with various numbers of LSTM layers to determine the ideal depth at which to extract significant patterns.
- Adjusting the number of convolutional layers to determine the ideal depth for a given set of data.
- Various combinations of stride and kernel size are used to capture pertinent temporal and spatial data.

By fine-tuning these extra hyperparameters, the hybrid model performs better. The performance of the hybrid model can be significantly increased by determining the appropriate amount of hyperparameters and fine-tuning them. These hyperparameters improve the effectiveness of feature selection and subsequently classification by preventing overfitting and underfitting and also reduce the computation cost by selecting only important features. To perform binary classification, the datasets were split into two groups: benign and assault. The dataset is classified as normal or attack as one kind of assault for multiclass classification, as Table 3 illustrates.

Table 3 Binary & Multiclass Classification

Dataset	Binary		Multiclass	
	No of records	Records type	No of records	Records type
WSN-DS	2	Normal or Attack	10	Blackhole, Grayhole
NSL-KDD	2	Normal or Attack	5	DoS, U2R, Probe, and R2L
UNSW-NB15	2	Normal or Attack	9	Shellcode, Backdoor, Generic
CIC-IDS 2017	2	Normal or Attack	7	Normal, Bot, Brute Force

Evaluation metrics

Recall, precision, F-measure, and Rand accuracy are examples of frequently used evaluation metrics that are biased and shouldn't be utilized unless the biases are clearly understood, along with the identification of the statistic's base case or chance levels. The authors of [55] go over several ideas and metrics that represent the likelihood that a forecast is accurate as opposed to random variation. Network traffic is categorized by intrusion detection systems as either regular or attacked. The amount of the total that is accurately measured as normal or under attack depends on accuracy. It serves as a starting point from which to adjust the parameters. In intrusion detection technologies, the confusion matrix is frequently utilized as a metric to assess classification performance. Accuracy refers to the ratio of correctly classified samples to the total number of samples, and its calculation formula is shown in Eq. 20.

$$Accuracy = \frac{TrueP + TrueN}{(TrueP + TrueN + FalseP + FalseN)} \tag{20}$$

To prevent needless notifications, it identifies the attacks that were real attacks. The PR formula is used to determine the percentage of correctly classified regular traffic samples among samples that are expected to be normal traffic and is given in Eq. 21.

$$Precision = \frac{TP}{(TP + FP)} \tag{21}$$

DR ascertains which actual attacks the model predicted. A high DR rate is more important since it can make the system sensitive to missed attacks. DR is the ratio of correctly recognized abnormal samples to expected abnormal sample counts. The method for calculating the DR, a crucial indicator in intrusion detection systems, is presented in Eq. 22. It represents the model's capacity to detect attacks.

$$DR = \frac{TP}{(TP + FN)} \tag{22}$$

It offers a fair assessment of memory and precision. The formula for calculating F1, the harmonic mean of PR and DR, is given in Eq. 23.

$$F1 = \frac{2 * (Precision * DR)}{(Precision + DR)} \tag{23}$$

Although it identifies the network traffic as normal, it is regarded as an attack. Low FPR is recommended to prevent alarms. The FPR calculation procedure is provided in Eq. 24. The false positive rate, or FPR, is defined as the ratio of incorrectly recognized abnormal samples to the expected number of normal samples.

$$FPR = \frac{FP}{(TP + FP)} \tag{24}$$

These measurements offer a thorough assessment of how well the model detects intrusions. The model uses detection rate and precision to find predictions that are reliable. Accuracy and false positive rate are used by the model to prevent improper classification. An overview of precision and detection rate is provided by the F1 score. These metrics offer a high-level perspective of the behaviour of the model, but there is a trade-off. A high DR is necessary if security is desired.

Results

Table 4 presents the results of the LSTM approach. The best model, which used 180 LSTM units distributed over hidden layers and a training time of 253.76 seconds, had a test accuracy of 89.26%. In its Dense layer, this model made use of the ReLU activation function. The best classifier in the GRU algorithm attained an F1 Score of 95.04%, a test accuracy of 86.10%, and a training time of 189.10 seconds. In its Dense layer, this model made use of the Softmax activation function. In addition, 120 GRU units were used in its hidden layers of configuration, and In Fig. 6, the outcomes demonstrate that compared to the more intricate GRU and LSTM algorithms, the RNN trains faster. But in comparison to the LSTM approach, the GRU approach trains more quickly. The Dense layer for every one of these techniques is calculated using the Softmax activation function. Every one of the five classes

Table 4 LSTM Binary Classification on NSL KDD

Hidden units	Activation function	Validation acc%	Test acc%	F1 score	Training (Seconds)
15	S2	97.16	84.36	96.38	62.41
15	R2	96.15	75.74	95.54	116.89
30	S2	97.80	82.19	96.89	142.06
30	R2	96.33	81.34	97.56	147.44
40	R2	95.50	82.23	95.67	152.82
40	S2	95.51	81.19	94.67	82.34
60	R2	94.55	84.98	95.28	124.44
60	S2	95.43	84.94	96.64	147.22
90	R2	95.28	83.80	97.46	164.51
90	S2	95.36	82.53	98.35	158.54
120	R2	94.63	81.38	97.80	270.41
120	S2	96.39	82.91	97.38	248.46
150	R2	97.49	81.13	98.58	215.46
150	S2	97.25	84.26	96.24	243.76
180	R2	96.49	85.13	98.58	235.46
180	S2	97.25	89.26	98.94	253.76

found in the NSL KDD dataset was taken into account when conducting the trials. The results for the RNN approach achieved an accuracy of 87.21%, an F1 score of 94.03%, and a validation accuracy of 95.93%. 180 RNN units were utilized in the hidden layers of this classifier, and it took 139.55 seconds to train. A model with 180 LSTM units spread over 5 layers achieved a test accuracy of 88.41%, an F1 score of 98.64%, and a validation accuracy of 98.25% when it came to the LSTM approach as shown in Table 5. The best classifier in the GRU algorithm example achieved an 88.55% test accuracy, a 95.61% F1 score, and a 152.17-second training time. As shown in Fig. 7, by using the NSL-KDD dataset's reduced feature vector, we ran simulations on it during the second stage of our experiment. We took into consideration the LSTM, GRU, and RNN algorithms. The outcomes for the binary classification scheme are shown in Table 6. RNN's dense layer's ReLU activation function yielded a training time of 85.32%, an F1 score of 89.48%, and an 88.71% test accuracy. Table 7 presents the LSTM classifier's findings. F1 score of 97.91%, validation accuracy of 98.25%, and test accuracy of 88.60% were attained by the best model.

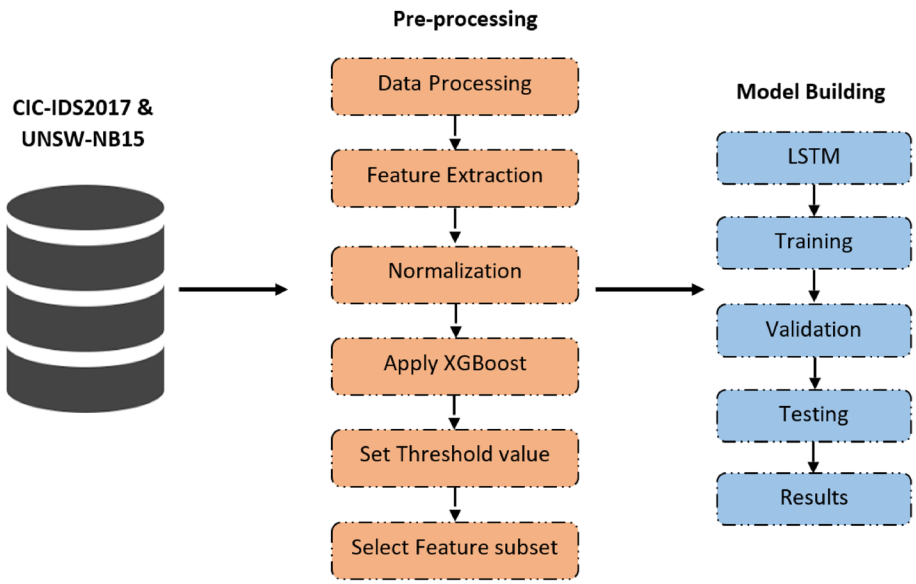

Fig. 6 Framework for IDS using XGBoost and LSTM

Table 5 LSTM Multiclass Classification on NSL KDD

Hidden units	Activation function	Validation acc%	Test acc%	F1 score	Training (Sec)
15	S2	88.16	82.36	96.38	73.41
15	R2	87.15	78.74	94.54	116.89
30	S2	90.80	83.19	96.89	172.06
30	R2	92.33	84.34	94.56	167.44
40	R2	93.50	83.23	96.67	152.82
40	S2	91.51	82.19	97.67	162.34
60	R2	95.55	85.98	94.28	144.44
60	S2	96.43	83.94	94.64	147.22
90	R2	94.28	81.80	93.46	134.51
90	S2	95.36	82.53	96.35	146.54
120	R2	93.63	81.38	97.80	268.41
120	S2	94.39	83.91	94.38	231.46
150	R2	96.49	87.13	96.58	235.46
150	S2	97.25	84.26	98.24	238.76
180	R2	97.49	87.13	98.58	255.46
180	S2	98.25	88.41	98.64	233.76

180 units were placed in the hidden layers of this model, and its Dense Layer was equipped with the ReLU activation algorithm. The results show that the most effective model, the RNN, obtained a test accuracy of 83.20%, an F1 score of 87.09%, and a validation accuracy of 88.77%. With 180 units in the hidden layers, this classifier yielded a training time of 158.65 seconds. The most effective model was trained in 195.85 seconds with 180 units in the hidden layers, and it achieved a test accuracy of 97.91% and a validation accuracy of 98.25% as shown in Table 7. The results provided by the best RNN model are not as good as those achieved by the LSTM in terms of test accuracy. Figures 8 compares the training time-frames of each model on the UNSW NB15 dataset for the RNN multiclass classification job. Figures 9 and 10 compare the training time-frames of each model on the NSL KDD and UNSW NB15 datasets respectively for the LSTM multiclass classification job. The trends show that the LSTM has the longest training period for a sophisticated RNN. In a subsequent stage, we selected two datasets-CIC IDS 2017 and WSN DS-to apply the CNN-LSTM model. The outcomes would then be compared to determine whether the dataset is more accurate in predicting intrusion detection. For this reason, we first looked at the

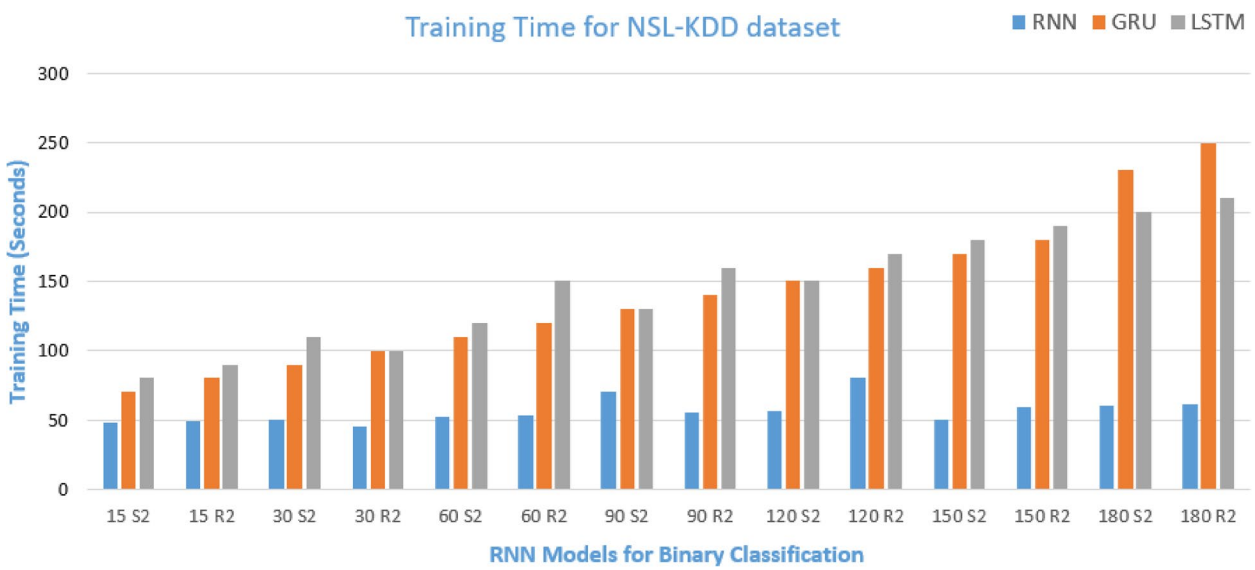

Fig. 7 RNN models Binary Classification on NSL KDD

Table 6 LSTM Binary Classification on UNSW NB15

Hidden units	Activation function	Validation acc%	Test acc%	F1 score	Training (Sec)
15	S2	87.16	89.36	86.38	72.41
15	R2	95.15	95.74	85.54	126.89
30	S2	93.80	92.19	86.89	132.06
30	R2	95.33	91.34	87.56	157.44
40	R2	94.50	92.23	85.67	122.82
40	S2	93.51	91.19	93.67	112.34
60	R2	92.55	94.98	92.28	134.44
60	S2	91.43	94.94	97.64	157.22
90	R2	90.28	93.80	92.46	174.51
90	S2	93.36	92.53	91.35	188.54
120	R2	98.63	91.38	93.80	220.41
120	S2	98.39	92.91	94.38	238.46
150	R2	96.49	91.13	93.58	205.46
150	S2	95.25	94.26	95.24	263.76
180	R2	98.49	96.13	96.58	225.46
180	S2	96.25	97.26	97.94	263.76

performance of datasets based on CNN, LSTM, and then CNN-LSTM to see which model yielded the best results. Tables 8 and 9 present the findings. The accuracy of the CIC IDS 2017 binary dataset is displayed in Table 8. Five-layer CNN-LSTM structures had the best accuracy at 98.55%, with five-layer LSTM structures coming in second at 97.95%. Lastly, three-layer CNN structures at 96.09%. WSN DS behaves differently, as seen in Table 9, the CNN-LSTM structure with a five-layer structure had the best accuracy at 97.35%. LSTM followed with five layers at 97.80% and two layers at 97.23% of CNN accuracy. We used the CNN-LSTM hybrid structure to carry out our research after comparing three learning algorithms. The selection of characteristics for the model-building process was the focus of our second testing session. The CIC IDS 2017 dataset and a single CNN-LSTM layer were used in our initial six trials, which included 28, 38, 45, 52, and 60 features. We examined 8, 14, and 20 features based on WSN DS. Table 10 presents the findings derived from the binary CIC IDS 2017 dataset. Twenty-eight characteristics had scores of 96.10% for detection rate and 95.65% for accuracy, respectively. The accuracy

Table 7 LSTM Multiclass Classification on UNSW NB15

Hidden units	Activation function	Validation acc%	Test acc%	F1 score	Training (Sec)
15	S2	88.16	82.36	96.38	73.41
15	R2	87.15	78.74	94.54	116.89
30	S2	90.80	83.19	96.89	172.06
30	R2	92.33	84.34	94.56	167.44
40	R2	93.50	83.23	96.67	152.82
40	S2	91.51	82.19	97.67	162.34
60	R2	95.55	85.98	94.28	144.44
60	S2	96.43	83.94	94.64	147.22
90	R2	94.28	81.80	93.46	134.51
90	S2	95.36	82.53	96.35	146.54
120	R2	93.63	81.38	97.80	268.41
120	S2	94.39	83.91	94.38	231.46
150	R2	96.49	87.13	96.58	235.46
150	S2	97.25	84.26	97.24	238.76
180	R2	97.49	87.13	97.58	255.46
180	S2	98.25	88.60	97.91	195.85

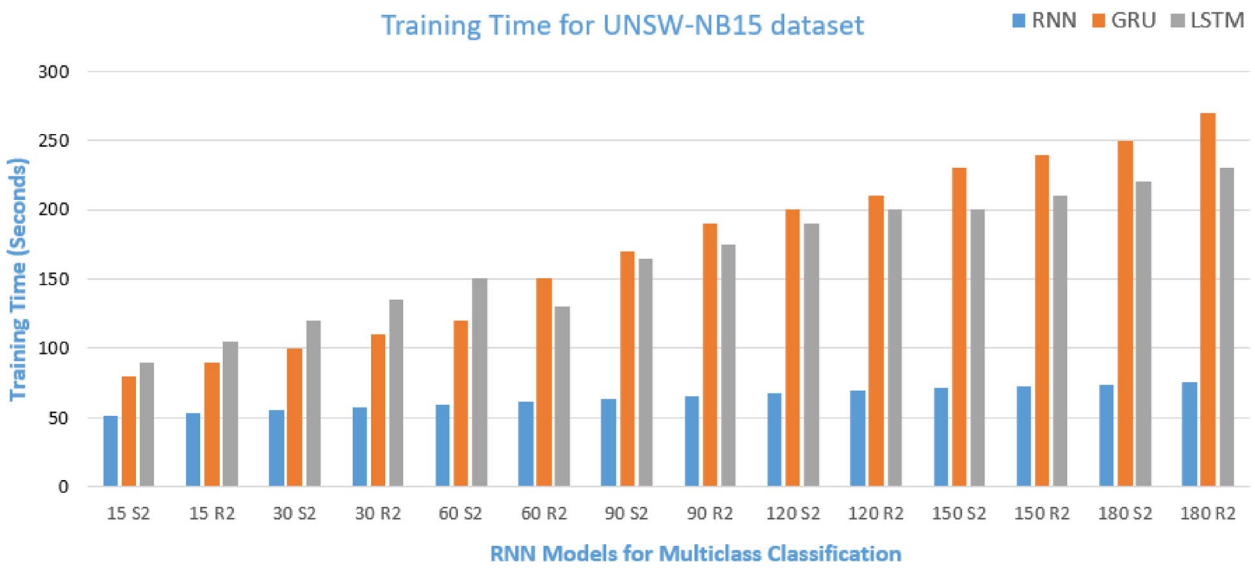

Fig. 8 RNN model Multiclass Classification on UNSW NB15

and detection rate of 38 features are 95.31% and 95.41%, respectively. The accuracy and detection rate for 45 features were 95.95 percent and 94.90 percent, respectively. The accuracy and detection rates for 52 features were 96.79% and 96.55%, respectively, but for 60 features they were 97.90 percent and 93.35 percent. Previous results showed that 52 features had the highest F1-score value, the lowest false alarm rate, and the highest detection rate, while 60 features had the maximum accuracy. Table 11 presents the findings derived from the binary WSN DS dataset. Out of the 42 features, 14 had the lowest FAR value. We also noticed that 42 features took less time to train the data than 32 features when we looked at the training time, therefore we kept experimenting with 42 features. Based on binary WSN DS, the final feature selection test was conducted. An analysis of 18 features identified a model's ideal performance. 18 features had accuracy and detection rates of 96.95 and 96.10%, respectively, while 14 and 8 features had detection rates of 91.10 and 95.10% and accuracy of 94.31 and 95.04%, respectively. Furthermore, the IDS model for this dataset was trained using the complete feature set (Fig. 10).

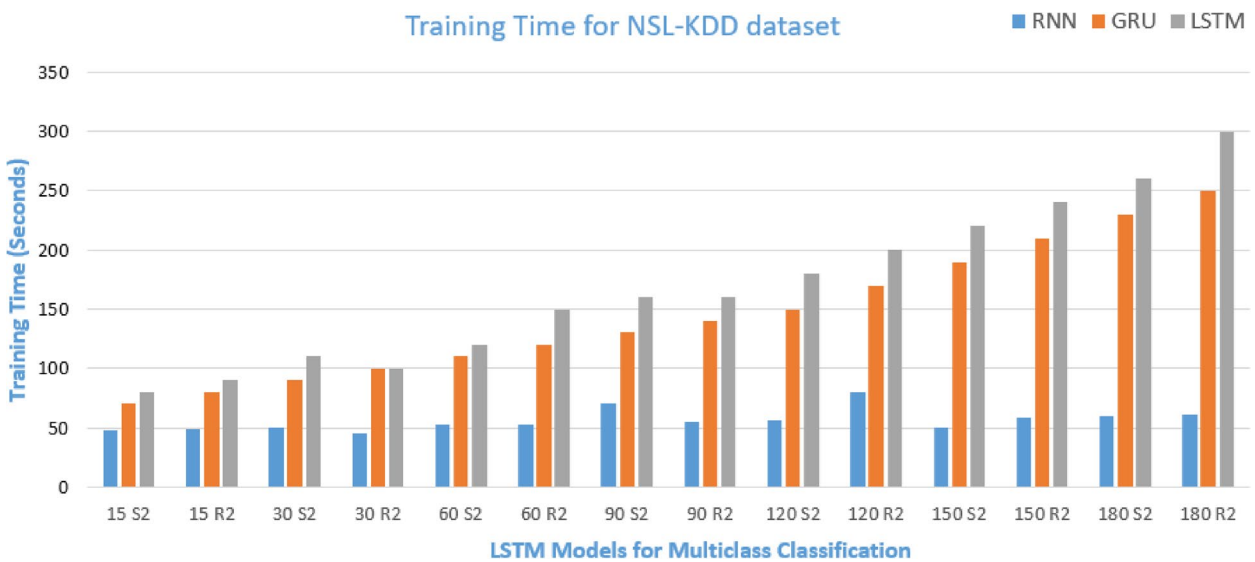

Fig. 9 LSTM model Multiclass Classification on NSL KDD

Table 8 Binary Classification Accuracy on CIC-IDS 2017

# of layers	CNN	LSTM	CNN-LSTM
1	96.54	97.65	94.09
2	97.23	97.21	97.31
3	96.09	96.35	96.90
4	96.76	97.80	97.85
5	96.90	97.95	98.55

Table 9 Binary Classification Accuracy on WSN DS

# of layers	CNN	LSTM	CNN-LSTM
1	95.04	96.65	97.10
2	96.31	96.21	97.41
3	96.95	97.35	96.90
4	96.79	97.80	97.55
5	96.90	97.05	97.35

Discussion

Datasets

The first step in building a successful intrusion detection system is to choose a suitable dataset. Both benign and malicious records should be included in the dataset to simulate the kinds of records the model will come across in the actual world. We use benchmark datasets CIC IDS 2017, UNSW NB15, NSL KDD, and WSN DS in this research. These datasets include both legitimate and malicious traffic data that is thought to be fresh and devoid of a sizable quantity of excess information.

Below are the characteristics of each dataset and how each class relates to the hybrid model:

CIC-IDS 2017 As indicated in Table 12, the CIC-IDS 2017 dataset contains eleven new threats, including brute force, port scan, denial of service, and online assaults like SQL Injection and XSS. It was developed in 2017 by the Canadian Institute for Cybersecurity and its eighty features are used to monitor communications that are harmful and benign [56].

- Features: The sole binary classes in the CIC-IDS 2017 dataset are normal and attack, which stand in for network packets. The most important feature of this dataset is that it includes the most recent attack types, including DoS, U2R, and R2L. Table 12 provides additional details.
- Relevance: Due to a mix of category and numerical characteristics. While XGBoost handles the categorical properties, LSTM finds the temporal correlations in the network's packet sequences. In contrast, CNNs retrieve information from the packets, like flow time and packet size, whereas LSTMs record successive patterns of attacks.

WSN DS To distinguish between benign and malicious communication, WSN DS was created in 2016 and uses sensors to track the number of nodes in wireless networks. The LEACH routing protocol is used to extract the dataset's records, which are represented by 23 features. Four different types of DoS attacks exist flooding, grayhole, blackhole, and TDMA [57].

- Features: This dataset includes Wireless Sensor Networks (WSNs). This dataset offers a distinct perspective on wireless network traffic and is primarily focused on dangers in WSNs.

Table 10 Binary Classification FS(%) on CIC-IDS 2017

# of layers	Acc	PR	DR	F1 score	FAR	Training (Sec)
28	94.04	95.65	96.10	94.10	0.17	653
38	95.31	95.21	95.41	95.10	0.23	376
45	95.95	94.35	94.90	94.10	0.81	540
52	96.79	95.80	96.55	97.10	0.10	432
60	97.90	96.05	93.35	95.10	0.53	1050

Table 11 Binary Classification FS(%) on WSN DS

# of layers	Acc	PR	DR	F1 score	FAR	Training (Sec)
8	95.04	94.65	95.10	95.10	2.30	70
14	94.31	96.21	96.41	94.10	0.90	90
20	96.95	95.35	97.90	96.10	3.90	140

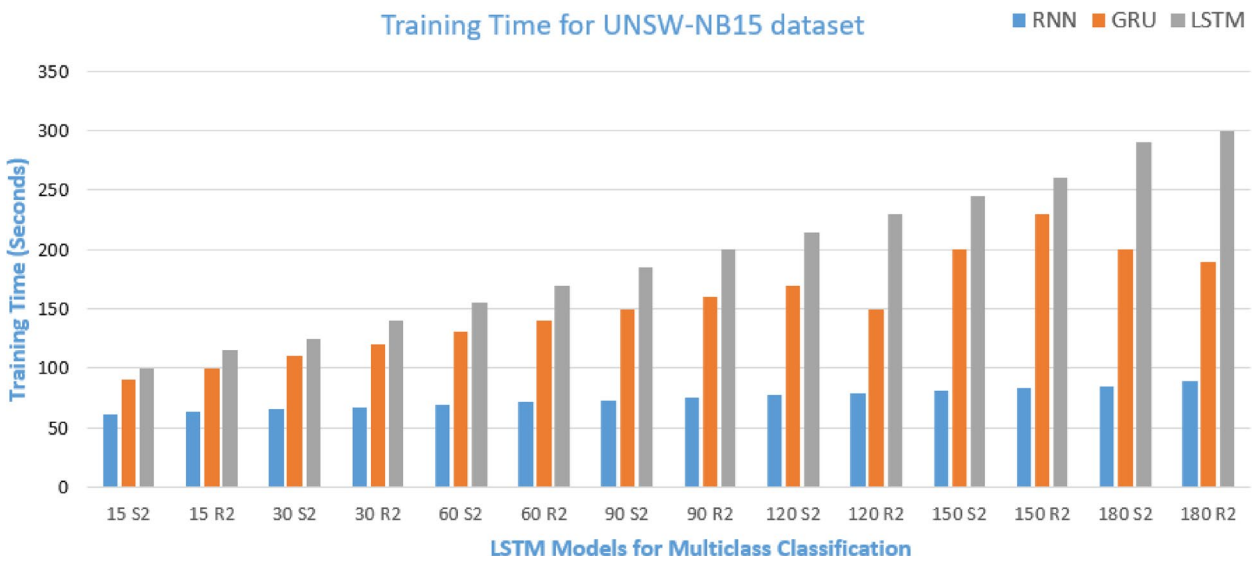

Fig. 10 LSTM model Multiclass Classification on UNSW NB15

Table 12 Dataset CIC-IDS 2017

Threat class	Dataset
Benign	2273097
Robot	1966
Distributed DoS	128027
Golden Eye (DoS)	10293
Hulk (DoS)	231073
Slowhttptest (DoS)	5499
DoSSlowloris	5796
FTP PATATOR	7938
HeartBleed	11
Infiltration	36
PortScan	158930
SSH PATATOR	5897
WebAttack BruteForce	1507
WebAttack SQL Injection	21
WebAttack XSS	652
Total	2830743

- Relevance: XGBoost manages many traffic types and extracts features pertinent to intrusion detection. On the other hand, the LSTM records sequential patterns of attacks while CNNs extract features from the packets, such as packet size and flow time.

NSL KDD In this analysis, the NSL KDD dataset [58] is considered. This is one of the datasets used for IDS framework evaluation the most frequently. In addition to regular network traffic, the NSL KDD also faces four additional types of intrusions: DoS, U2R, R2L, and Probe. This study considers two subsets of the NSL KDD dataset: the NSL KDD-Train and the NSL KDD Test+. Overfitting of the models during the training phase will be prevented by the validation technique [59]. The NSL KDD Test+ is used to test the validated models. Every feature in the NSL KDD dataset is listed in Table 13, with three features being categorical and the rest being numerical.

- Features: This dataset is somewhat antiquated. We have utilized this dataset because it has been used by other authors over the years. It includes both redundant data and a variety of fictitious instances.
- Relevance: Some features are irrelevant, and there is noise in the data. It helps pick up some crucial details. Further details are given in the Table 13

UNSW NB15 The UNSW NB15 contains 42 attributes, made up of 39 number inputs and 3 category inputs [60].

Table 13 Dataset NSL KDD

Class	Training	Weight %	Testing	Weight %
Normal	67342	44.16	9710	41.70
DoS	45927	42.40	7460	43.90
Probing	11656	10.15	2421	10.65
R2L	995	2.20	2885	2.75
U2R	52	1.09	67	1.00
Total	125972	100	22543	100

For the numerical inputs, the following data formats are applicable: binary, integer, and float. The UNSW NB15 also has two data subsets: one for testing and one for the training phase. Furthermore, the UNSW NB15-Test is used to test the verified models. These nine assault classifications are listed in Table 14.

- Features: UNSW-NB15 consists of contemporary network assaults. Table 14 lists the features that are present in this dataset.
- Relevance: As a result of a combination of numerical and category qualities. Whereas LSTM discovers the temporal connections in the packet sequences of the network, XGBoost manages the categorical characteristics. On the other hand, the LSTM records sequential patterns of attacks while CNNs extract features from the packets, such as packet size and flow time.

Edge_IIoT Ferrag et al. [61] presented the Edge_IIoT dataset given in Table 15, a novel cybersecurity dataset for IoT and IoT applications, as a realistic cybersecurity dataset for IoT environments in 2022. With 14 different assault kinds, it has 62 features. The 14 unique attacks against IoT and IIoT protocols in this dataset are divided into five categories based on the type of threat they pose: malware, injection, man-in-the-middle, distributed and denial-of-service, information gathering, and injection. To keep the percentages constant throughout all classes, we employed a stratification option. We set aside 85% of the sample for training and 15% for testing. After using CNN to extract features, we used SMOTE to address data imbalance. Using the Edge_IIoT dataset, we conducted training and testing and discovered that using binary classification improves performance. It assigned a test accuracy of 95.21% and classified the traffic on the Internet of Things as either normal or an assault. The project's main focus is building an efficient IDS that can discriminate between harmful and legitimate communications. The everyday discovery of new assaults has made cybersecurity concerns more difficult [29]. Furthermore, the high false alarm rate of conventional intrusion detection systems exposes the system to various types of assaults and prevents security analysts from identifying potentially dangerous ones. Due to duplicate information and obsolete data, the intrusion system training and evaluation procedure is subpar, resulting in inadequate training [30, 62]. Researchers have recently created IDS based on DL. A recent study found that deep learning outperforms classical learning methods in classifying received traffic in massive datasets, constantly attacking settings, and detecting fraudulent traffic [63, 64]. XGBoost, CNN, and LSTM configurations have been used in numerous research; the distinction is that they were done so independently. Because we blended the three algorithms XGBoost and CNN for feature extraction and selection and then applied these features to LSTM in both cases to produce a hybrid model XGBoost-LSTM and CNN-LSTM for our model, the data will be processed by both XGBoost and CNN at each stage. First, we conducted our analysis using the CNN and LSTM algorithms. We were able to produce a model with a high detection rate and accuracy thanks to the five layers of hybrid CNN and LSTM in the model's structure [34]. On datasets, preprocessing operations such as encoding, data normalization, and feature selection for model training were carried out [65]. The output was supplied to CNN's first layer, which extracted spatial features, the LSTM layer, which extracted temporal features, and the FC layer, which performed classification. Throughout 70 epochs, with 97.90% accuracy

Table 14 Dataset UNSW NB15

Class	Training	Weight %	Testing	Weight %
Normal	56000	34.10	37000	33.70
Generic	40000	28.20	18871	23.90
Exploits	33393	22.25	11132	16.65
Fuzzers	18184	6.40	6062	9.75
DoS	12264	3.15	4089	4.70
Reconnaissance	10491	2.20	3496	2.10
Backdoors	1746	1.10	583	3.70
Shellcode	1133	0.25	378	2.65
Analysis	2000	1.20	677	1.75
Worms	130	1.15	44	1.10
Total	175341	100	82332	100

Table 15 Dataset Edge_IIoT

Class	IoT traffic	No. of instances	Weight%
Normal	Normal	1,615,643	72.80291
Attack	DDoS UDP	121,568	5.478008
	DDoS ICMP	116,436	5.246753
	SQL injection	51,203	2.307272
	Password	50,153	2.259958
	Vulnerability scanner	50,110	2.25802
	DDoS TCP	50,062	2.255857
	DDoS HTTP	49,911	2.249053
	Backdoor	24,862	1.120313
	Port Scanning	22,564	1.016762
	XSS	15,915	0.71715
	Ransomware	10,925	0.492294

for binary classification and 98.40% accuracy for multiclass classification, CIC IDS 2017 earned the maximum accuracy as compared to [66]. Simultaneously, the multiclass classification yielded 98.38% and 97.43% precision and F1-scores, respectively, and the binary classification produced 97.90% accuracy. Based on the binary classification for the CIC IDS 2017 and WSN DS datasets, the lowest FPR was found to be between 0.10% and 0.90%.

In this research, the proposed method is compared with the existing binary classification in Table 16, which represents the second phase of the proposed hybrid model. For the NSL KDD dataset, the XGBoost-LSTM achieved a test accuracy of 94.41%. This performance was better than that of other approaches that were already in use. The XGBoost-GRU achieved a test accuracy of 90.72% for the UNSW NB15. This result was better than what was produced using the techniques in [32, 35]. A comparison between the techniques suggested in this study and those reviewed in the literature is shown in Table 17. We enumerated and cited the pertinent methods. The Feature Selection Technique is also included in the table where applicable. Furthermore, the comparative analysis's performance indicator is the multiclass classification method's test accuracy over the UNSW NB15 and NSL KDD datasets. The findings demonstrate that in the case of the NSL KDD datasets, the XGBoost-LSTM (90.71%) and CNN-LSTM (91.09%) algorithms outperformed other current approaches [67, 68]. Compared to other approaches, the MCA-LSTM (89.25%) strategy performed ideally for the UNSW NB15 dataset [69]. Furthermore, we analyzed the models that outperformed the UNSW NB15 and NSL KDD (XGBoost-LSTM) and the CIC-IDS 2017 and WSN DS (CNN-LSTM) in terms of multiclass classification accuracy. When compared to the models suggested in earlier research, the suggested NIDS-CNNLSTM model has considerably enhanced performance in terms of ACC, DR, and FPR, as indicated by the comparison results in Tables 16 and 17. The model presented in this research significantly lowers the false positive rate while significantly increasing the intrusion detection model's detection rate and accuracy by taking into account a wide range of assessment indicators. This finding was also verified using other datasets, which could have varied outcomes in the end because the training and test sets were chosen at random. It can still demonstrate, nevertheless, that the suggested model is better than the ones that have already been put forth in earlier research. Additionally, it can adjust to network attack traffic in a variety of situations with greater assurance. The intrusion detection system performs better overall when efficiency is increased while maintaining accuracy in the intrusion detection scenario. We calculated a Confusion Matrix (CM) for every model, which enabled us to evaluate the models' performance on different classes within the datasets. Class0 = Normal, class1 = R2L, class2 = U2R, class3 = Probe, and class4 = DoS for the dataset NSL KDD. Class 0= Normal, Class1 = Generic, Class2 = Exploits, Class3 = Fuzzers, Class4 = DoS, Class5 = Reconnaissance, Class6 = Analysis, Class7 = Backdoor, Class8 = Shellcode, and Class9 = Worms for the dataset UNSW NB15. The confusion matrix shows how effective the XGBoost-LSTM was at class 0, class 1, class 3, and class detection. It was able to anticipate class 1 attacks, but it was unable to identify class 2 attacks. This is because, in the NSL KDD dataset, the U2R incursion is the minority class. The examination of the XGBoost-LSTM performance on some classes from the UNSW NB15 dataset shows that class 0-5 and class 7-9 could benefit from this strategy.

Table 16 Binary Classification - Comparison with SOTA Methods

Model	Dataset	Feature selection	Test accuracy
[32] MCA-LSTM	UNSW NB15	IG	95.21
[Current] RNN	UNSW NB15	XGBoost	93.10
[32] MCA-LSTM	NSL KDD	IG	94.12
[67] LSTM	NSL KDD	XGBoost	91.57
[35] RNN	UNSW NB15	-	92.19
[Current] XGBoost-LSTM	UNSW NB15	XGBoost	94.41
[Current] CNN-LSTM	WSN DS	CNN	94.18
[Current] CNN-LSTM	Edge_IIoT	CNN	95.21

Table 17 Multiclass Classification - Comparison SOTA Methods

Model	Dataset	Feature selection	Test accuracy
[32] MCA-LSTM	UNSW NB15	IG	87.14
[Current] RNN	UNSW NB15	XGBoost	84.79
[32] MCA-LSTM	NSL KDD	IG	89.25
[67] LSTM	NSL KDD	-	88.18
[35] RNN	UNSW NB15	XGBoost	85.39
[Current] XGBoost-LSTM	UNSW NB15	CNN	90.71
[Current] CNN-LSTM	WSN DS	-	91.09

Comparison with latest IDS techniques

We discovered the following outcomes after contrasting our hybrid model with the most recent intrusion detection methods and are given in Table 18. In a study [18], the IGEA algorithm was employed with the UNSW-NB15 dataset to detect intrusions with an accuracy of 80.40%. Conversely, [19, 20, 22] achieved nearly

Table 18 Comparison with Latest IDS Techniques

Model	Dataset	Algorithm	Accuracy%
[18]	UNSW-NB15	IGEA	80.40
[19]	CICIDS2017	LSTM-AE	99.00
[20]	KDDCUP99	XGBoost	99.99
[21]	CICIDS-2017	CNN-GRU	98.73
[22]	Edge_IIoT	CNNBiLSTM	99.64
[24]	BoT-IoT	RF-AE	99.63
[25]	ToN-IoT	OCNN-LSTM	94.40
[3]	AWD	Radial based Function	98.68
[70]	CIC2017	Gradient Boosting	94.09
[71]	NSL KDD	LSTM-RNN	88.40
Current	UNSW-NB15,NSL-KDD	CNN-LSTM, XGBoost-LSTM	98.40

99% while using various datasets for their intrusion detection system. Using CNN, XGBoost, and LSTM, we applied the UNSW-NB15, NSL-KDD, and CICIDS2017 datasets and obtained an accuracy of 98.40 as compared to our hybrid model. Using the same datasets, our model outperformed [25, 70, 71] and obtained an accuracy of 98.40 as opposed to 94.40, 94.09, and 88.40, respectively. Some of the models outperformed our hybrid model by 0.60, and the authors used more recent datasets than we did. Our hybrid model was not performed with contemporary datasets; this will be done in the future.

Ablation study

We evaluated the datasets based on CNN, LSTM, LSTM-CNN, and CNN-LSTM in the ablation study phase of our research to check which model produced the greatest results. The results are shown in Tables 19, 20, 21. The accuracy of the CIC-IDS2017 binary dataset is displayed in Table 19. The highest accuracy of 96.21% was attained by CNN-LSTM structures with 5 layers, and 96.05% was attained by LSTM-CNN structures with 5 layers. Lastly, 3 layers CNN at 95.15% and LSTM structures at 95.63%. Table 20 displays the UNSW-NB binary dataset results. The maximum accuracy was obtained by 5 layers of CNN-LSTM at 95.95%, then 4 layers of CNN-LSTM at 95.82%, 5 layers of CNN at 95.10%, and 4 layers of CNN at 94.85%. WSN-DS behaves differently. As indicated in Table 21, the LSTM-CNN structure with a 5-layer structure had the best accuracy at 94.95%. CNN-LSTM and LSTM followed with 94.95% for 5 layers and 94.80% for 5 layers of CNN. We used the CNN-LSTM hybrid structure to carry out our research after comparing four learning algorithms.

Table 20 Accuracy of UNSW-NB15 Binary Classification

# of layers	CNN	LSTM	LSTM-CNN	CNN-LSTM
1	93.11	93.29	93.40	94.02
2	94.52	93.80	93.70	94.25
3	94.65	94.10	94.40	94.50
4	94.85	94.60	94.70	95.82
5	95.10	94.89	95.15	95.95

Error analysis

The hybrid model XGBoost-LSTM and CNN-LSTM intrusion detection system is susceptible to errors and misclassifications, which can throw the model out of balance. There are a few prevalent problems with potential solutions One important element that misclassifies benign traffic as harmful is false positives. Our model learns too much during training and is unable to generalize traffic that has not been encountered. The dataset may be unbalanced, which would raise the false positive rate. To prevent overfitting, we used the regularisation procedure. To help the model discover new patterns and prevent dataset imbalance, network traffic data is synthesized through the process of augmentation. To detect intrusions, attacks that are incorrectly identified can be given higher weights. When there are false negatives, our model fails to detect real intrusions because of feature selection bias, class imbalance, and zero-day attacks. To find new invasions, we trained the model using adversarial instances. To enhance feature selection and more effectively detect intrusion patterns, we used CNN and XGBoost. We used CNN-LSTM and XGBoost-LSTM to improve the model's robustness. Network traffic data intrusions don't happen frequently. In the intrusion

Table 19 Accuracy of CIC-IDS2017 Binary Classification

# of layers	CNN	LSTM	LSTM-CNN	CNN-LSTM
1	94.12	94.19	94.40	95.02
2	94.55	94.30	94.60	95.25
3	95.15	94.51	95.49	95.50
4	95.35	95.09	95.80	95.82
5	95.60	95.63	96.05	96.21

Table 21 Accuracy of WSN-DS Binary Classification

# of layers	CNN	LSTM	LSTM-CNN	CNN-LSTM
1	93.31	93.19	93.50	94.20
2	94.42	93.30	93.55	94.35
3	94.55	94.15	93.65	94.40
4	94.75	94.30	94.74	94.62
5	94.80	94.89	94.95	94.89

detection systems, there may be an imbalance in classes. We can employ SMOTE, oversampling, or undersampling approaches to balance the dataset. SMOTE was employed in our model to address the problem of network traffic data imbalance. Methods for artificially balancing the class distribution in the training data include oversampling and undersampling. Another issue with deep learning models is explainability, which makes it challenging to comprehend why certain classifications, such as false positives or negatives, are made. Explainability techniques or attention mechanisms can be used to refute the internal logic underlying the methodical classification process.

Scalability and real-world applications

The scalability of the proposed hybrid model can be increased by using GPUs or TPUs. Additionally, it can be enhanced by combining XGBoost with other algorithms that take less resources and are computationally efficient. As an example, in this hybrid model, we have used XGBoost-LSTM, which is computationally efficient. The hybrid XGBoost-LSTM and CNN-LSTM models can be scaled using diffident factors. LSTM and CNN are computationally expensive and require more hardware resources. However, XGBoost performs better and is scalable for large datasets. The hybrid model's ability to extract features from XGBoost and its ability to use LSTM to identify temporal correlations in sequences make it applicable to real-world scenarios such as stock price prediction and weather forecasting. Using a hybrid model architecture, anomalous patterns in fraud detection or network traffic analysis can be found. CNN-LSTM's feature extraction capabilities and ability to capture sequential information in phrases make it useful for sentiment analysis, topic modeling, and other applications.

Challenges and potential overcoming strategies

Given that computing cost is a key consideration, using the hybrid model in a large-scale network context presents certain obstacles. More resources are needed for XGBoost-LSTM and CNN-LSTM since they generate large amounts of data through intricate computations. Whether to scale the hybrid model vertically or horizontally presents another challenge. Real-time network applications require low latency at all times. The model's performance may be impacted by delays in the network traffic processing. Another problem with certain predictions is interpretability, which limits the choices taken in mission-critical applications because it is unclear how they are made. The suggested hybrid model deployment may be complicated by the need to train the model and then fine-tune the hyperparameters. Several strategies can be used to overcome the obstacles encountered in the deployment of the proposed hybrid model: more robust feature selection techniques, such as XGBoost, can be employed to extract the most relevant features; hyperparameters, the most important component of the model, can be carefully tuned to reduce processing requirements and help optimize the model for accuracy and efficiency; distributed frameworks can be used to train the model using multiple machines, which will enhance scalability and accelerate training; GPUs or TPUs can be used to enhance processing; and occasionally, trade-off strategies can be used to benefit from various approaches. We used the aforementioned strategies to get over the obstacles in the way of the suggested hybrid model's deployment.

Conclusion

The volume of data transferred across communication infrastructures has increased recently due to advancements in technologies such as cloud computing, the Internet of Things (IoT), automobile networks, etc. Network systems are used to transmit diverse and heterogeneous data in dispersed environments as communication technology develops. Attackers, on the other hand, have increased their efforts in an attempt to render systems on networks susceptible. An efficient intrusion detection system is now essential since hackers are always creating new kinds of attacks and networks are getting bigger. In this paper, a hybrid model combining deep learning and machine learning techniques for intrusion detection is implemented. XGBoost, CNN, and LSTM are the three types of machine and deep learning techniques used in this study. We employed the CNN-LSTM model with the CIC-IDS 2017 and WSN DS datasets, and the XGBoost-LSTM model with the UNSW NB15 and NSL KDD datasets. To address the issue of data imbalance, we suggest a hybrid sampling and deep network-based approach for network intrusion detection systems. Furthermore, as the feature dimension increases, current IDSs have trouble identifying new threats due to low test accuracy scores. The findings showed that XGBoost with LSTM slightly performed better than the CNN-LSTM model. These findings also showed that, in contrast to alternative methods, our hybrid models executed more effectively and operated at peak efficiency. There are still several areas that need to be improved, despite our hybrid model outperforming a few SOTA approaches in terms of accuracy and detection rate and performs very well when using IDS datasets for training and testing. In the future, we will employ updated datasets to detect the newest threats in the network, such as zero-day attacks, instead of using the hybrid mode on some outdated datasets used to detect intrusions. Attack patterns need to be looked into further, and the collected data should

be used to improve detection. Despite its benefits, the suggested model requires more training time due to its complexity than standard methods such as using a single model. Future research aims to improve the model's performance on minority classes and explore how well the suggested framework performs on specific classes found in the datasets under study. While the suggested method has a quick detection speed, By addressing the model's low detection rate and high false alarm rate, which are brought on by the imbalanced records in the dataset, we hope to improve its performance in the future.

Supporting information
All relevant data is within the manuscript and its Supporting Information files.

Authors' contributions
Conceptualization, M.S., T.S.M. and K.R.M.; Data Gathering, A.A., T.S.M, and A.R.; Formal analysis, A.A., A.H.K. and J.T.; Funding acquisition, A.A.; Investigation, J.T., M.S., and K.R.M.; Methodology, M.S., T.S.M., and A.H.K.; Supervision, K.R.M., and T.S.M.; Writing—original draft, A.H.K., A.R., and M.S.; Writing—review & editing, A.R., A.A., and J.T. All authors have read and agreed to the published version of the manuscript.

Funding
This work was supported by King Saud University, Riyadh, Saudi Arabia, through Researchers Supporting Project number RSP2024R184.

Availability of data and materials
Data is provided within the manuscript or supplementary information files.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 8 June 2024 Accepted: 6 July 2024
Published online: 17 July 2024

References

1. Deebak BD, Hwang SO (2024) "Healthcare Applications Using Blockchain With a Cloud-Assisted Decentralized Privacy-Preserving Framework," in IEEE Transactions on Mobile Computing. 23(5):5897–916. https://doi.org/10.1109/TMC.2023.3315510.
2. Ahmad Z, Shahid Khan A, Wai Shiang C, Abdullah J, Ahmad F (2021) Network intrusion detection system: a systematic study of machine learning and deep learning approaches. Trans Emerg Telecommun Technol 32:e4150
3. Heidari A, Navimipour NJ, Unal M (2023) A secure intrusion detection platform using blockchain and radial basis function neural networks for internet of drones. IEEE Internet Things J. 10:8445–54
4. Chkirbene Z, Erbad A, Hamila R, Mohamed A, Guizani M, Hamdi M (2020) Tidcs: A dynamic intrusion detection and classification system based feature selection. IEEE Access 8:95864–95877
5. Junwon K, Jiho S, Ki-Woong P, Jung TS (2022) "Improving Method of Anomaly Detection Performance for Industrial IoT Environment". Computers, Materials & Continua. 72(3):5377–94. https://doi.org/10.32604/cmc.2022.026619.
6. Hassan SR, Rehman AU, Alsharabi N, Arain S, Quddus A, Hamam H (2024) Design of load-aware resource allocation or heterogeneous fog computing systems. PeerJ Comput Sci. 10:e1986. https://doi.org/10.7717/peerj-cs.1986.
7. Heidari A, Jafari Navimipour N, Dag H, Unal M (2024) Deepfake detection using deep learning methods: A systematic and comprehensive review. Wiley Interdiscip Rev Data Min Knowl Disc 14:e1520
8. Halbouni A, Gunawan TS, Habaebi MH, Halbouni M, Kartiwi M, Ahmad R (2022) Cnn-lstm: hybrid deep neural network for network intrusion detection system. IEEE Access 10:99837–99849
9. Molina-Coronado B, Mori U, Mendiburu A, Miguel-Alonso J (2020) Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process. IEEE Trans Netw Serv Manag 17(4):2451–2479
10. Heidari A, Jafari Navimipour N, Unal M, Zhang G (2023) Machine learning applications in internet-of-drones: Systematic review, recent deployments, and open issues. ACM Comput Surv 55(12):1–45
11. Bukhari SMS, Zafar MH, Abou Houran M, Moosavi SKR, Mansoor M, Muaaz M, Sanfilippo F (2024) Secure and privacy-preserving intrusion detection in wireless sensor networks: Federated learning with scnn-bi-lstm for enhanced reliability. Ad Hoc Netw 155(103):407
12. Hanafi AV, Ghaffari A, Rezaei H, Valipour A, Arasteh B (2024) Intrusion detection in internet of things using improved binary golden jackal optimization algorithm and lstm. Clust Comput 27(3):2673–2690
13. Belouch M, hadaj SE (2017) Comparison of ensemble learning methods applied to network intrusion detection. ACM, pp 1–4
14. Wu P (2020) Deep learning for network intrusion detection: Attack recognition with computational intelligence. PhD thesis, UNSW Sydney
15. Quinlan JR (2014) C4. 5: programs for machine learning. Elsevier
16. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press
17. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press
18. Vashishtha LK, Singh AP, Chatterjee K (2023) Hidm: A hybrid intrusion detection model for cloud based systems. Wirel Pers Commun 128:2637–2666
19. Hnamte V, Nhung-Nguyen H, Hussain J, Hwa-Kim Y (2023) A novel two-stage deep learning model for network intrusion detection: Lstm-ae. IEEE Access
20. Talukder MA, Hasan KF, Islam MM, Uddin MA, Akhter A, Yousuf MA, Alharbi F, Moni MA (2023) A dependable hybrid machine learning model for network intrusion detection. J Inf Secur Appl 72:103405
21. Henry A, Gautam S, Khanna S, Rabie K, Shongwe T, Bhattacharya P, Sharma B, Chowdhury S (2023) Composition of hybrid deep learning model and feature optimization for intrusion detection system. Sensors 23(2):890
22. Hnamte V, Hussain J (2023) Dcnnbilstm: An efficient hybrid deep learning-based intrusion detection system. Telematics Inform Rep 10:100053
23. Mohamed D, Ismael O (2023) Enhancement of an iot hybrid intrusion detection system based on fog-to-cloud computing. J Cloud Comput 12(1):41
24. Wang C, Sun Y, Wang W, Liu H, Wang B (2023) Hybrid intrusion detection system based on combination of random forest and autoencoder. Symmetry 15(3):568
25. Lilhore UK, Manoharan P, Simaiya S, Alroobaea R, Alsafyani M, Baqasah AM, Dalal S, Sharma A, Raahemifar K (2023) Hidm: Hybrid intrusion detection model for industry 4.0 networks using an optimized cnn-lstm with transfer learning. Sensors 23(18):7856
26. Mehmood M, Javed T, Nebhen J, Abbas S, Abid R, Bojja GR, Rizwan M (2022) A hybrid approach for network intrusion detection. CMC-Comput Mater Contin 70:91–107
27. Muhammad SF, Sagheer A, Atta-ur-Rahman, Kiran S, Muhammad AK, Amir M (2022) "A Fused Machine Learning Approach for Intrusion Detection System" Computers, Materials & Continua. 74(2):2607–23. https://doi.org/10.32604/cmc.2023.032617.
28. Mebawondu JO, Alowolodu OD, Mebawondu JO, Adetunmbi AO (2020) Network intrusion detection system using supervised learning paradigm. Sci Afr 9:e00497
29. Nedeljkovic D, Jakovljevic Z (2022) Cnn based method for the development of cyber-attacks detection algorithms in industrial control systems. Comput Secur 114:102585

30. Shi WC, Sun HM (2020) Deepbot: a time-based botnet detection with deep learning. Soft Comput 24:16605–16616
31. Kaya Ç, Yıldız O, Ay S (2016) Performance analysis of machine learning techniques in intrusion detection. In: 2016 24th Signal Processing and Communication Application Conference (SIU), IEEE, pp 1473–1476
32. Dong RH, Li XY, Zhang QY, Yuan H (2020) Network intrusion detection model based on multivariate correlation analysis-long short-time memory network. IET Inf Secur 14(2):166–174
33. Injadat M, Moubayed A, Nassif AB, Shami A (2020) Multi-stage optimized machine learning framework for network intrusion detection. IEEE Trans Netw Serv Manag 18(2):1803–1816
34. Zhang H, Li Y, Lv Z, Sangaiah AK, Huang T (2020) A real-time and ubiquitous network attack detection based on deep belief network and support vector machine. IEEE/CAA J Autom Sin 7(3):790–799
35. Yin C, Zhu Y, Fei J, He X (2017) A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5:21954–21961
36. Kasongo SM, Sun Y (2020) A deep learning method with wrapper based feature extraction for wireless intrusion detection system. Comput Secur 92:101752
37. Kolias C, Kambourakis G, Stavrou A, Gritzalis S (2015) Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. IEEE Commun Surv Tutor 18(1):184–208
38. Javaid A, Niyaz Q, Sun W, Alam M (2016) A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). ACM, New York City. pp 21–26
39. Yan B, Han G (2018) Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. IEEE Access 6:41238–41248
40. Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. IEEE Trans Emerg Top Comput Intell 2(1):41–50
41. Ieracitano C, Adeel A, Morabito FC, Hussain A (2020) A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. Neurocomputing 387:51–62
42. Kim JY, Bu SJ, Cho SB (2017) Malware detection using deep transferred generative adversarial networks. In: Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part I 24, Springer, pp 556–564
43. Shahriar MH, Haque NI, Rahman MA, Alonso M (2020) G-ids: Generative adversarial networks assisted intrusion detection system. In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), IEEE, pp 376–385
44. Yilmaz I, Masum R, Siraj A (2020) Addressing imbalanced data problem with generative adversarial network for intrusion detection. In: 2020 IEEE 21st international conference on information reuse and integration for data science (IRI), IEEE, pp 25–30
45. Dlamini G, Fahim M (2021) Dgm: a data generative model to improve minority class presence in anomaly detection domain. Neural Comput & Applic 33:13635–13646
46. Nakatsukasa Y, Soma T, Uschmajew A (2017) Finding a low-rank basis in a matrix subspace. Math Program 162:325–361
47. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, ACM, New York City. pp 785–794
48. Gumus M, Kiran MS (2017) Crude oil price forecasting using xgboost. In: 2017 International conference on computer science and engineering (UBMK), IEEE, pp 1100–1103
49. Ren X, Guo H, Li S, Wang S, Li J (2017) A novel image classification method with cnn-xgboost model. In: Digital Forensics and Watermarking: 16th International Workshop, IWDW 2017, Magdeburg, Germany, August 23-25, 2017, Proceedings 16, Springer, pp 378–390
50. Amaouche S, Guezzaz A, Benkirane S, Azrour M (2024) Ids-xgbfs: a smart intrusion detection system using xgboostwith recent feature selection for vanet safety. Clust Comput 27:3521–3535
51. Putchala M (2017) Deep learning approach for intrusion detection system 687 (IDS) in the internet of things (IoT) network using gated recurrent neural 688 networks (GRU). PhD thesis, MS thesis, Dept. Comput. Sci. Eng., Wright State 689 Univ., Dayton
52. Naseer S, Saleem Y, Khalid S, Bashir MK, Han J, Iqbal MM, Han K (2018) Enhanced network anomaly detection based on deep neural networks. IEEE Access 6:48231–48246
53. Agarap AF (2018) Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375
54. Zhu D, Lu S, Wang M, Lin J, Wang Z (2020) Efficient precision-adjustable architecture for softmax function in deep learning. IEEE Trans Circ Syst II Expr Briefs 67(12):3382–3386
55. Powers DM (2020) Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061
56. Khan ZI, Afzal MM, Shamsi KN (2024) A comprehensive study on cic-ids2017 dataset for intrusion detection systems. Int Res J Adv Eng Hub 2(02):254–260
57. Almomani I, Al-Kasasbeh B, Al-Akhras M, et al (2016) Wsn-ds: A dataset for intrusion detection systems in wireless sensor networks. J Sens 2016:4731953
58. NSL-KDD (2024) The nsl-kdd : Nsl kdd dataset. https://www.unb.ca/cic/datasets/nsl.html. Accessed 14 May 2024.
59. Cawley GC, Talbot NL (2010) On over-fitting in model selection and subsequent selection bias in performance evaluation. J Mach Learn Res 11:2079–2107
60. Moustafa N, Slay J (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 military communications and information systems conference (MilCIS), IEEE, pp 1–6
61. Ferrag MA, Friha O, Hamouda D, Maglaras L, Janicke H (2022) Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning. IEEE Access 10:40281–40306
62. Chakravarthi B, Ng SC, Ezilarasan M, Leung MF (2022) Eeg-based emotion recognition using hybrid cnn and lstm classification. Front Comput Neurosci 16:1019776
63. Yao R, Wang N, Liu Z, Chen P, Sheng X (2021) Intrusion detection system in the advanced metering infrastructure: a cross-layer feature-fusion cnn-lstm-based approach. Sensors 21(2):626
64. Maseer ZK, Yusof R, Bahaman N, Mostafa SA, Foozy CFM (2021) Benchmarking of machine learning for anomaly based intrusion detection systems in the cicids2017 dataset. IEEE Access 9:22351–22370
65. Sun P, Liu P, Li Q, Liu C, Lu X, Hao R, Chen J (2020) Dl-ids: Extracting features using cnn-lstm hybrid network for intrusion detection system. Secur Commun Netw 2020:1–11
66. Al S, Dener M (2021) Stl-hdl: A new hybrid network intrusion detection system for imbalanced dataset on big data environment. Comput Secur 110(102):435
67. Altulaihan E, Almaiah MA, Aljughaiman A (2024) Anomaly detection ids for detecting dos attacks in iot networks based on machine learning algorithms. Sensors 24(2):713
68. Nazir A, He J, Zhu N, Qureshi SS, Qureshi SU, Ullah F, Wajahat A, Pathan MS (2024) A deep learning-based novel hybrid cnn-lstm architecture for efficient detection of threats in the iot ecosystem. Ain Shams Eng J 15:102777
69. Tang TA, Mhamdi L, McLernon D, Zaidi SAR, Ghogho M (2016) Deep learning approach for network intrusion detection in software defined networking. In: 2016 international conference on wireless networks and mobile communications (WINCOM), IEEE, pp 258–263
70. Kim T, Pak W (2022) Early detection of network intrusions using a GAN-based one-class classifier. IEEE Access 10:119357–119367
71. Ibrahim M, Elhafiz R (2023) Modeling an intrusion detection using recurrent neural networks. J Eng Res 11:100013

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.