

Intrusion detection in a controlled computer network environment using hybridized random forest and long short-term memory algorithms

A. I. Bassey^{ID*}, M. A. Agana, E. A. Edim, O. Njama-Abang

Department of Computer Science, University of Calabar, Calabar, Nigeria

Abstract

The unending reliant on network access in everything we do, and with the increasing dominance of online communication, there is urgent need to address computer network security challenges now, more than ever. Most of the existing Intrusion Detection Systems (IDSs) struggle to keep pace with the ever-changing characteristics of newly emerging threats. This research proposes a hybrid model that is effective in detecting both majority and minority attack classes using Random Forest (RF) and Long Short-Term Memory (LSTM) algorithms in a controlled computer network environment. Individually, the RF and LSTM models have limitations, but their individual strengths were extracted and subsequently hybridized to cover each other's weaknesses. To handle the challenge of class imbalance, a class weight was applied to the model's loss function. This approach prompts the model to give extra attention to the minority class attacks. The meta classifier optimized the RF-LSTM combination and offered a more improved model that is effective in detecting both majority and minority classes simultaneously. The hybrid model was analyzed using the Neural Simulator Language- Knowledge Discovery in Databases (NSL-KDD) dataset. The model was deployed in a virtual network environment consisting of three operating systems and a host. The RF – LSTM hybrid model performed exceptionally well by achieving a prediction accuracy of 98.3%, precision of 96.98, recall and F1score of 96.58 and 97.03 respectively, all after 100 epochs at 0.01 learning rate. This outcome addresses shortcomings and lapses hitherto suffered by most intrusion detection models (assessing minority class attacks).

DOI:10.46481/asr.2025.4.3.327

Keywords: Intrusion, Detection, Security, Hybrid

Article History :

Received: 02 June 2025

Received in revised form: 07 October 2025

Accepted for publication: 13 October 2025

Published: 01 December 2025

© 2025 The Author(s). Published by the [Nigerian Society of Physical Sciences](#) under the terms of the [Creative Commons Attribution 4.0 International license](#). Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

1. Introduction

The significant increase in network-based services as well as the sensitive information they deal with has necessitated the urgency in addressing network security now more than ever. According to Khraisat *et al.* [1] attackers are employing more advanced and intricate methods, and the primary difficulty lies in detecting malware that is both unfamiliar and intentionally made obscure. Given the widespread actions of hackers in recent years, it becomes increasingly essential to have an intrusion detection system to secure the network to avoid information theft, leakage and misuse. According to Ref. [2], an intrusion detection system (IDS) safeguards

*Corresponding author Tel. No.: +234-703-527-4272.

Email address: ahenabassey01@gmail.com (A. I. Bassey^{ID})

networks by keeping an eye on data flow, identifying unusual patterns and reporting such. IDS's primary objective is to alert the system administrator to any unusual activity. To complement and improve upon the conventional security methods, the Machine Learning and deep Learning methods are often adopted by most researchers. These techniques have proven brilliance in achieving high detection accuracy.

The task of an Intrusion Detection System (IDS) is generally to classify network behavior as normal or abnormal. IDS are classified into two major categories: signature-based and anomaly detection. Misuse detection attempts to discover malicious activities based on the information of known attacks. Therefore, detection takes place if the monitored activities resemble the signature of a known attack. Tesfahun *et al.* [3] considered it to be effective in dealing with known attacks but struggles with novel attacks. Nguyen & Reddi. [4] explained that the Deep learning methods especially, are well known for their proficiency in handling both marked and unlabeled data, use the powerful Graphic Processing Unit (GPU) to solve complicated problems. In addition, attackers most times continue to develop new tools and methods to take advantage of vulnerabilities of all kinds. Hence, detecting every kind of assault with a single set method is quite challenging. As a result, Intrusion detection systems (IDS) are becoming a crucial component of network security. It is used to monitor traffics on a network and send alerts when any attacks occur. Ahmed *et al.* [5], also explained that examining User to Root (U2R), Remote to Local (R2L), probing, and Denial-of-Service (DoS) attacks in networks simultaneously has been a major challenge in intrusion detection. The fact remains that computer users cannot forecast which sort of assaults to expect. It is prudent to be ready for all types of attacks, even though one class may be more frequent than the others as explained by Ref. [6]. In view of the above identified security lapses, this research work fills the gaps by adopting a hybrid approach of both RF and LSTM algorithms that improves network security by functioning consistently across all attack classes. Farnaaz & Jabbar. [7] in their research identified Random Forest (RF) as an ensemble classifier that is best suited for classifying the intrusion attacks with higher accuracy in intrusion detection. Yin *et al.* [8] also identified LSTM as an outstanding deep learning method that does much better in extracting representation from the data to create more suitable models for detecting intrusion. Intrusion detection systems (IDSs) are another security method that businesses utilize to protect their networks. The ensemble nature of RF is of great benefits to the hybrid model because the aggregated decisions from the multiple decisions trees help reduce the impact of noisy data and increase overall robustness. RF also provides insights into feature importance, helping us understand which features contributes most to the detection of intrusions. LSTM on the other hand facilitated real-time detection by capturing temporal dependencies in network traffic. For applications requiring timely responses to security threats, the hybridized strengths of these two models are most suitable. Also, the meta classifier deployed to optimized the combination ensured a more accurate final prediction is achieved.

In view of the above, the proposed hybrid model aims at developing a detection system that is efficient in assessing both majority and minority attack classes simultaneously. This hybrid approach leverages the strengths of both models to address the limitations of using a single model. The main contributions of this paper are as follows:

- (i) The RF-LSTM hybrid model classifies network traffic more accurately by leveraging RF's classification ability on the processed data and LSTM's sequence-aware learning.
- (ii) False positives is reduced through the RF's ability to filter out irrelevant features. While LSTM's ability to recognize temporal patterns in sequential data enhances the detection of complex, time-based attacks that might otherwise be missed, consequently reducing false negatives.
- (iii) Explores how different Machine Learning and Deep Learning based algorithms can be jointly combined.
- (iv) The study also fosters further advancement and improvement by setting a new criterion for future studies in the area of intrusion detection. Other researchers can use this as a baseline to compare their new models.

2. Related works

Several techniques based on Machine Learning (ML) and Deep Learning (DL) have emerged in recent decades.

Prasanna *et al.* [9] proposed "A Convolution Neural Network (CNN) and Long Short-Term Memory (LSTM) Model for Intrusion Detection System from High Dimensional Data" and developed a model that worked well in detecting Denial-of-Service (DoS) and Probing attacks, the paper suggested a system by which CNN and LSTM algorithms were merged. The experimental data suggested that the merged model increased the precision of detection of human influence and boost the efficiency of the detection approach. However, it was less efficient in detecting minority class attacks like Remote to Local (R2L) and User to Root (U2R). Therefore, if most of the threats on the network in the future are U2R and R2L, this kind of IDS might not work very well.

Devulapalli [6] in the work titled: "A Machine Learning Approach for Uniform Intrusion Detection" adopted the Ensemble models approach; a process by which two or more machine learning models were combined to produce a model with higher predictive power. Using the Multilayer Perceptron (MLP), the J48 Decision Tree with Synthetic Minority Oversampling Technique (SMOTE) was combined. By adopting the up-sampling technique on the minority class, the recall increased to 97.60%. Vinayakumar *et al.* [10] revealed that in their work, they employed a number of DL algorithms for IDS. They employed CNN-LSTM, Convolution Neural Network (CNN), Recurrent Neural Network (RNN), Convolution Neural Network (CNN) and Gated Recurrent Unit (GRU)

combination models. When tested on the KDDCup '99 dataset, the CNN-LSTM combination model achieves an accuracy score of 99.7% in binary classification and 98.7% in multi-class classification. However, the models struggled to assess the minority class attacks.

Bediako [11] suggested an LSTM-RNN-based Distributed Denial of Service (DDoS) detector. The Central Processing Unit (CPU) and Graphic Processing Unit (GPU) were both used to test the LSTM-RNN's performance. The trials were conducted using the NSL-KDD dataset. 99.968% was the noteworthy detection accuracy. The research reveals that hostile threats arise and evolve constantly; consequently, the network requires a very good security solution. Owing to the intricacy of novel attacks, most existing intrusion detection models are incapable of detecting attacks.

Hammad [12] developed a cyber-assault detection machine by combining the strengths of Random Forest and Long Short-Term Memory algorithms. The proposed RF-LSTM hybrid model was evaluated using real-global IoT healthcare datasets. The integration of both models resulted in an increased accuracy of 97% in real time scenarios. The model was developed specifically for DDoS attacks, its ability to assess minority attack types was not part of the consideration of the research study. Qazi *et al.* [13] constructed a deep learning-based hybrid intrusion detection system using a convolutional recurrent neural network. To determine the effectiveness of this hybrid model, experiments were conducted using the publicly accessible benchmark Canadian Institute for Cyber Security Intrusion Detection System (CICIDS-2018) dataset. Result showed that the proposed HDLNIDS outperformed the selected intrusion detection approaches with an average accuracy of 98.90% in detecting malicious attacks. A CNN-LSTM hybrid model as proposed by Bamber *et al.* [14] was successfully trained on the NSL-KDD dataset. Data was preprocessed with Recursive Feature Elimination (RFE) and a Decision Tree classifier in order to identify the most important features. The effectiveness of the CNN-LSTM hybrid model was proven as it outperformed other Deep Learning models like ANN, LSTM, BiLSTM, GRU and BiGRU during evaluation with an accuracy of 95%, recall of 0.89 and F1 score of 0.94. Also, a deep learning solution for detecting attacks was implemented by Laghrissi *et al.* [15], two techniques; the Principal Component Analysis (PCA) and Mutual Information (MI) were deployed for dimensional reduction. Result showed that models based on PCA achieved the best accuracy for training and testing in both binary and multiclass classification.

An advanced Intrusion Detection System (IDS) that used a hybrid Long Short-Term Memory FeedForward (LSTM-FF) Neural Network was designed by Omer *et al.* [16] to improve detection, automated feature selection using Random Forest was incorporated. This approach performed highly in multi-class classification with a 99.54% accuracy rate, 93.19% precision, 99.59% specificity and 90.28% F1 score. Xue *et al.* [17] introduced a hybrid autoencoder with an enhanced LSTM-CNN architecture. Three datasets; NSL-KDD, UNSW-NB15 and CICIDS-2018 were deployed to evaluate the performance of the model, it recorded 95.7%, 94.9% and 96.7% accuracy for each respectively. Azar *et al.* [18] proposed four models namely; Random Forest(RF), Long Short-Term Memory (LSTM), Artificial Neural Networks (ANN) and Gated Recurrent Unit (GRU). Two datasets; Satellite Terrestrial Integrated Networks (STINs) and UNSW-NB15 which simulates terrestrial networks were deployed to evaluate the performance of the proposed SAT-IDS. All the proposed systems adopted the Sequential Forward Features Selection approach based on random forest to select important features from the dataset. The RF-SFS-GRU model exhibited the highest accuracy (79%) across the three proposed hybrid deep learning based SAT-IDS. Halbouni *et al.* [19] designed a hybrid intrusion detection system with high detective capability. Based on the ability of CNN to extract spatial features and the strength of LSTM in extracting temporal features The model was trained using three datasets; CICIDS-2017, UNSW-NB15 and WSN-DS. The hybrid model displayed high accuracy and a relatively low False Alarm Rate (FAR). Sajid *et al.* [20] also proposed a hybrid Extreme Gradient Boosting (XGBoost) and Convolutional Neural Networks (CNN) for feature extraction and further merged each of these algorithms with LSTM for classification. The result showed a high detection rate, high accuracy with relatively low False alarm rate, which proves the usefulness of the hybrid model. S. Meftah *et al.* [21] applied two stage anomaly-based network intrusion detection process using UNSW-NB15 dataset. Adopting some data mining techniques like Logistic Regression (LR), Support Vector Machine (SVM) and Gradient Boost, classification was performed. The experimental result recorded an accuracy of 86.04%.

Most existing IDS are effective in detecting DoS and Probe attacks but struggle to capture minority class attacks like U2R and R2L. However, the oversampling technique is being adopted by most research works. This approach has a specific disadvantage because the model learns the specific characteristics of these duplicates or synthetically generated data points instead of learning the general underlying patterns of the minority attack class, the model actually memorizes the training data but fails to detect new or slightly different attack instances when deployed on real-world data. This over fitting problems reduces the ability of the model to detect novel or evolving attacks, since the system is only effective in identifying the specific attacks, it memorized. But an unfolding variant of the same attacks could pass unnoticed.

To address this gap, this research work adopted a technique of applying class weights to the model's loss function without actually altering the original dataset via resampling, this method prompts the model to put more efforts into correctly identifying the minority attack types, since a single error on a minority sample now contributes much more to the total loss than an error on a majority sample. This implies that the model now assumes that it is much more costly to miss a minority attack than to miss-classify normal traffic.

This technique was implemented by setting the class weight parameter on TensorFlow. The 'balance' option which automatically calculates weight inversely proportional to class frequencies was used. For example:

$$\omega_1 = N_{\text{total}} / (N_{\text{class}} \times N_1), \quad (1)$$

where ω_i is the weight for class i , N_{total} is the total number of sample in the dataset, N_{class} is the number of unique classes, N_i is the number of samples in class i .

In view of the above identified lapses of some existing IDSs, this research work is aimed at developing a hybrid model that is effective in detecting both majority and minority attack classes using Random Forest (RF) and Long Short-Term Memory (LSTM) algorithms in a controlled computer network environment, so that network security is improved.

2.1. Intrusion detection systems (IDSs)

An Intrusion detection system (IDS) is an evolving security measure that offers excellent protection for information held within network systems. The primary aim of implementing the IDS is to identify abuse, unauthorized usage, and exploitation of network system attacks and to thwart their execution. According to Devi *et al.* [22], the fundamental goal of IDS is to notify the system administrator if any suspicious behavior occurs. Intrusion detection systems (IDS) can generally be divided into two main categories: anomaly approach and misuse method. The capacity to identify assaults based on predetermined indicators of harmful activity is known as the misuse approach. Systems utilize patterns or signatures of known attacks to compare collected data or actual behaviors. Ozkan-Okay *et al.* [23], explained that the IDSs examine the gathered data and compare it to attack patterns stored in a large database of known assaults in order to detect signatures. For the anomaly detection approach, system administrators establish the normal profile (baseline) for the network traffic, protocols, and typical package size, breakdown, and attempt to detect traffic on deviations created by normal network behavior.

2.2. Basic functions of IDSs

According to Ref. [23], IDS technologies vary widely in terms of the kinds of attacks they can identify and the methods they employ to identify them. All IDS types must have some unique features in addition to the capacity to record and analyze events in order to identify undesired ones. The following functionalities were considered:

2.2.1. Information Recording

Retaining information locally may be helpful for constructing and comparing profiles that are normally set. Furthermore, the recorded data is transmitted independently to management systems, information security programs and central recording servers.

2.2.2. Important events identification

It is essential to promptly and precisely identify a condition that deviates from the data that is routinely reported and regarded as typical.

2.2.3. Identified important events being notified

These alerts, also known as notifications, are sent via a variety of channels, including emails and messages in the system's user interface. Typically, a notice includes basic details concerning suspicious events that have happened. To find out more, system users must access the IDS.

2.2.4. Reports generation

The system reports that are created either give a thorough account of noteworthy events or summarize observed events. For instance, IDS can gather more specific data if it detects suspicious activities during the session. It can also alter parameters like, when alerts should be sent out following the detection of a threat. The fundamental characteristics shared by all IDS types is its inability to offer a totally accurate detection. When an IDS interprets a typical behavior as an attack, it is known as a false positive. If it is unable to see and detect harmful behavior as usual, it is a false negative. Eliminating all these false positives and negatives is not feasible. Actually, lowering one usually results in an increase in the other. Even if the false positive rate rises, many IDS developers would rather lower the false negative rate.

2.3. Network intrusion detection

This software is strategically placed throughout networks to passively examine traffic moving through the devices they are installed on. NIDS can be software or hardware-based systems, and they can connect to a variety of network media, including Ethernet and Fibre Distributed Data Interface (FDDI), depending on the system's manufacturer. In order to listen in promiscuous mode to network talks, NIDS have two networks' interfaces. The other interface is utilized for reporting and control. According to Ahmad *et al.* [24], NIDS are positioned at one or more important points in order to monitor traffic to and from all devices on the network. The entire subnet's passing traffic is analyzed, and the traffic is compared to the library of known attacks. Once an assault is discovered, or aberrant behavior is sensed, the warning can be forwarded to the administrator. Installing an NIDS on the subnet where firewalls are situated to detect attempts to breach the firewall is an example of an NIDS. Scanning all incoming and outgoing data is ideal, but doing so could result in a bottleneck that slows down the network as a whole. Because they are passive, NIDS don't alter the traffic they are monitoring.

2.4. NIDS' security features

Although network-based intrusion detection systems (IDSs) provide a multitude of security features, they can be generically categorized into three types:

2.4.1. Collection of information

The ability of network-based intrusion detection systems to collect data from communication networks is limited. Information about associated hosts and network activity is typically gathered. The following is a list of some of the information aspects that were gathered:

- (i) Finding Hosts: A list of network hosts can be generated by an IDS.
- (ii) Operating System Identification: It is possible to identify the operating systems and versions that hosts are using. Finding susceptible hosts is made easier with knowledge about the operating systems versions being utilized.
- (iii) Applications Identification: By keeping an eye on open ports and application communication, an IDS sensor can determine the version of an application. Applications that may be vulnerable and their unauthorized use are identified using this information.
- (iv) Identifying Network Characterization: Broad details on traffic, network configuration, and a few IDS sensors are gathered.

2.4.2. Logging

IDSs that are network-based record detailed information about events that are detected. This information is utilized for incident correlation, investigation and alert validation. The following data categories are frequently logged by network-based IDSs:

- (i) Time and date.
- (ii) Quantity of connections.
- (iii) Event kind.
- (iv) Procedures.
- (v) IP addresses of the source and the destination.
- (vi) Quantity of packets sent.
- (vii) Requests and answers for applications.

2.4.3. Detection

To perform detailed analysis and increase the detection rate, many network-based intrusion detection systems (IDSs) combine signature-based and anomaly-based methods. When the anomaly-based method detects unusual activity, it breaks it down into requests and responses that are then compared with the signatures of known attacks; in other words, the methods are implemented in a nested manner.

3. Methodology

This research adopted a machine learning development life cycle approach with the primary objective of hybridizing Random Forest algorithm and Long Short-Term Memory as well as leveraging analytics processes on the dataset provided. Some selected process from the machine learning developmental life cycle adopted for this research study include (i) collection of data (ii) preprocessing of data (iii) model development/building (iv) implementation stage (v) evaluation of model and (vi) model deployment.

3.1. Data collection

The dataset used in this research is the NSL-KDD with 42 attributes present in every record. One element indicates the type of attack, while 41 attributes reflect the distinctive features of the data.

The distribution of all data contained in the NSL-KDD dataset is displayed in Table 1. It also captures the quantity and ratio of both training set and test set data.

Table 1 shows the percentage ratio of train and test sets. The train set constitute about 80% while the test set constitutes 20% of the total dataset. Also normal data makes up about 53.46% of the entire dataset, majority attack classes (DoS and Probe) makes up 45.7% of the attacks, while the minority classes make ups just 0.83%.

Table 1: Distribution of NSL-KDD dataset.

Type of label	NSL KDD (100%)		Training set (80%)		Test set (20%)	
	Quantity	Ratio (%)	Quantity	Ratio (%)	Quantity	Ratio (%)
Normal	67343	53.46	60659	53.50	6684	53.05
DoS	45927	36.46	41323	36.45	4604	36.55
Probing	11656	9.25	10461	9.23	1195	9.49
R2L	995	0.79	884	0.78	111	0.88
U2L	52	0.04	48	0.04	4	0.03
Total	125973	100	113375	100	12598	100

3.2. Data preprocessing

Before feeding the data into the RF and LSTM models, the following activities were carried out:

- (i) Loading the dataset into the data analysis tool (Python with pandas)
- (ii) Transforming categorical features into numerical format using one-hot-encoding technique. This technique was adopted because there is no inherent order to the categories, each category is treated as an independent feature to avoid misinterpretation of categorical data encoded with as single integer. Also, this technique allows the model to capture more complex relationships in the data.

e.g. assign values to the protocol type as: TCP = 1 UDP = 0 ICMP = 0,

UDP = 0 TCP = 1 ICMP = 0,

ICMP = 0 UDP = 0 TCP = 1.

The binary values are the one-hot encoded representation of TCP, UDP and ICMP. 1 indicates the presence of that category for a given data point, and 0 indicates its absence.

- (iii) Normalizing numeric features using the min-max scaling technique: This scales the values of each feature to a specific range, (typically between 0 and 1)

Mathematically,

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad (2)$$

where x_{scaled} = the scaled value of the feature, x = the original value of the feature, $x_{\min}$ = the minimum value of the feature in the dataset, $x_{\max}$ = the maximum value of the feature in the dataset.

For example, for a feature duration with minimum value of 0 and a maximum value of 100,000, a value of 50,000 would scale as: $x_{\text{scaled}} = (50000 - 0)/(100000 - 0) = 0.5$.

Each feature's values are scaled to a predetermined range, usually between 0 and 1. Applying appropriate normalization techniques to both numeric and categorical features, ensured every feature made an equal contribution to the machine learning model.

- (iv) Splitting the datasets into training and testing sets using the stratified splitting method which is most appropriate for imbalanced datasets. This ensured that the training and testing sets have a near same proportion of each class as the original dataset in order to prevent the possibility of neglect of the minority class during training. The splitting was in the ratio, 8:2 for training and testing sets respectively.

Training of the model took place over 50, 70 and 100 epochs respectively, with 125,973 training samples and 22,544 samples for validation and testing. To test the model's prediction over the training and test data, a lower learning rate of 0.01 was adopted after the initial learning rate of 0.1 was used to train the model.

- (v) Assigning higher weights to underrepresented classes during model training.

Mathematically, weight of class is:

$$i = \frac{N}{\text{num of samples in class } i}, \quad (3)$$

where N = total number of samples in the dataset, num of samples in class i = number of samples in class i .

- (vi) Hyper-parameter tuning process: The random search was adopted to find optimal combination of values that facilitates the model's best performance. The rationale behind this choice is because some hyper-parameters have greater impact on performance than others, so random search is more likely to explore a wider range of the search space and identify good solution faster.

3.3. Feature extraction

The principal component analysis technique (PCA) was adopted to reduce dimensionality of the dataset. By lowering the number of dimensions of the feature space and then extracting a subspace that best characterizes the data, the PCA produced new uncorrelated variables that progressively maximize variance and lower the computing costs and the error of parameter estimate.

Technically, after standardizing the data, PCA extracts the eigenvectors and eigenvalues from the covariance matrix (CM):

- (i) Calculate the covariance matrix of the preprocessed dataset:

$$\text{Cov}(X) = \frac{1}{n-1} \cdot (X - \mu)^T \cdot (X - \mu), \quad (4)$$

where $\text{Cov}(X)$ = the covariance matrix of the dataset X , n = number of samples, μ = mean vector of the dataset, T = transformed data matrix, $(X - \mu)$ = centered data matrix obtained by subtracting the mean from each data point.

- (ii) Calculate the eigenvalues and eigenvectors of the covariance matrix:

$$C \cdot V = \lambda \cdot V, \quad (5)$$

where λ = eigen values and v = eigen vectors.

- (iii) Sort the eigen values in descending order.

- (iv) The corresponding eigenvectors represent the principal component in order of importance.

- (v) Select the number of components that captures 95% of the total variance.

- (vi) Project the original data onto the selected principal component.

$$Y = X \cdot P, \quad (6)$$

where Y = the transformed data matrix, X = original data matrix, P = the matrix of selected eigen vectors.

Next, the PCA builds the projection matrix P from the selected k eigen vectors. Finally, it transforms the original dataset X via P to a k dimensional feature subspace.

3.4. Model building stage

The requirements found in the analysis served as the foundation for the models' development. Developing a blueprint for the new hybrid system is the main goal of the design phase.

3.5. System design

The logical design of the system shows how its components are arranged to complete a particular function. It focuses on the structure and organization of components. The final prediction is made by majority vote from all individual decision trees in the random forest ensemble:

$$Y_{\text{RF}}(x_i) = \text{mode}(\text{DT}_1(x_i), \text{DT}_2(x_i), \dots, \text{DT}_n(x_i)), \quad (7)$$

where n = number of trees in the RF, $\text{Mode}()$ = function that finds the most frequent class label among the prediction of all DTs.

The output of a LSTM unit is the hidden state, (a combination of the output gate (o_t) and the cell state (C_t)):

$$h_t = o_t * \tanh(C_t). \quad (8)$$

Hence, the output vector of new hidden state:

$$V_t = W_x * h_t + b. \quad (9)$$

Applying softmax function:

$$Y_t = \text{softmax}(V_t), \quad (10)$$

where o_t = output gate, W_x = weight matrix for respective gates and cell state, $\tanh$ = hyperbolic tangent activation function, b_x = biases for the respective gates (x), $*$ = vector point wise multiplication, Y_t = final output value for LSTM, C_t = current/updated cell state, h_t = hidden state at current time stamp (t), V_t = output of the new hidden state at time step (t).

Equations (7) and (10) are the final output equations for RF and LSTM, respectively. Using the late fusion approach, combine the final outputs from both models equations (7) and (10) using a function (Φ) to produce the final prediction (Y_{hybrid}):

$$\Phi(Y_{\text{RF}}(x_i), Y_t) = \left(\sum_i^2 W_{\text{RF}} * Y_{\text{RF}}(x_i) + W_{\text{LSTM}} * Y_t \right), \quad (11)$$

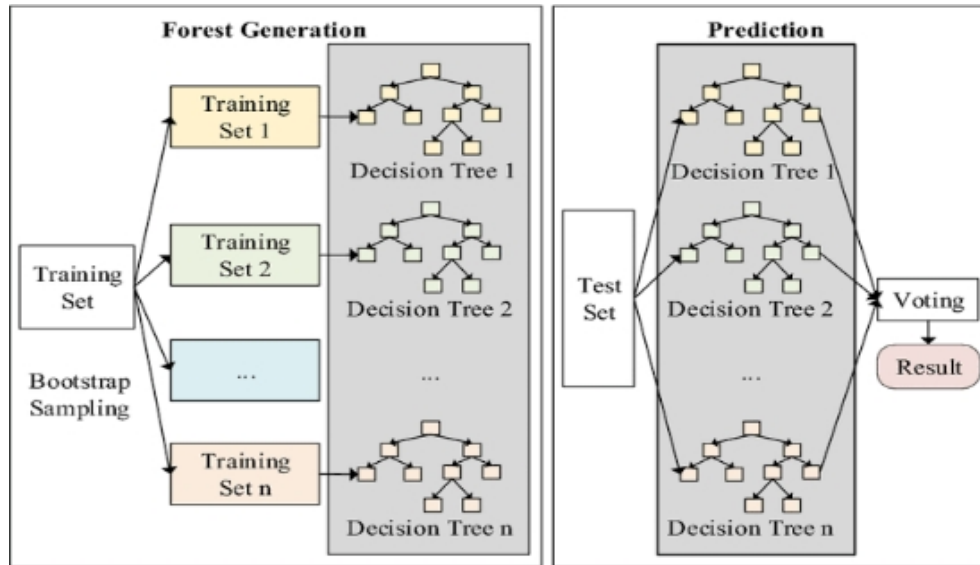


Figure 1: Random forest architectures [25].

$$Y_{\text{hybrid}} = W_{\text{RF}} * Y_{\text{RF}}(x_i) + W_{\text{LSTM}} * Y, \quad (12)$$

where Φ = a simple weighting average (trained to combine the final predictions of both RF and LSTM models), W_{RF} = assigned weight of random forest, $Y_{\text{RF}}(x_i)$ = final prediction of random forest, W_{LSTM} = assigned weight of LSTM, Y_i = final prediction of LSTM model.

The hybrid RF-LSTM model typically combines the strengths of both algorithms to enhance performance. The high efficiency of random forest in handling high-dimensional data and select relevant features was leveraged upon. While on the other hand, LSTM's capacity to learn temporal patterns and dependencies in sequential network traffic data was also considered. This informed the choice of this combinations.

3.5.1. Choice of system architecture

This research study adopted the parallel or ensemble architectural design method. In this design, both the RF and LSTM models are trained independently, and their outputs are combined via the meta classifier to make a final decision. In this case, another RF was deployed to learn the best way to combine the predictions of the base models. This method leverages the strengths of both models to create a more resilient system.

This approach improves the overall robustness of the system because if one model fails to detect a specific attack type, the other may succeed, thereby reducing the overall false negative rate.

3.5.2. Architecture of random forest model

The RF model is composed of different decision trees, each with the same nodes, but using different data that leads to different leaves as shown in Figure 1. The nodes are recursively split until a stopping criterion is met (e.g., max. depth & min. sample per leaf). The different decision trees are trained on different subsets of the training data and through bootstrapping technique, the data points are sampled with replacement from the original dataset. This creates new training sets with some data points appearing multiple times and others omitted entirely. This technique helps the ensemble learn from different aspects of the data. Once the model is trained, the test set is fed into it, the model makes prediction (classification) for each data point in the test set. Each leaf node is then assigned a class label, representing the majority class of the data points reaching that leaf. The final prediction is made by majority vote from all individual decision trees in the ensemble. The soft voting approach was adopted to aggregate the predictions of individual trees. It is a more accurate approach that considers the confidence of each prediction and better suited for scenarios with multiple classes or uncertain predictions on imbalance datasets. The RF algorithm leverages the strengths of multiple decision trees, each with some level of randomness, to improve the overall classification accuracy and robustness in detecting intrusions within the NSL-KDD dataset.

3.5.3. Architecture of the LSTM model

From Figure 2, it can be seen that the hidden state is an output of the LSTM cell used for prediction. It contains the information of previous inputs (from cell state) along with current input. It carries the output to the last cell (short term memory). The cell state is aggregated with all the past data information and is the long-term information retainer. Inputs provided to the LSTM are fed into

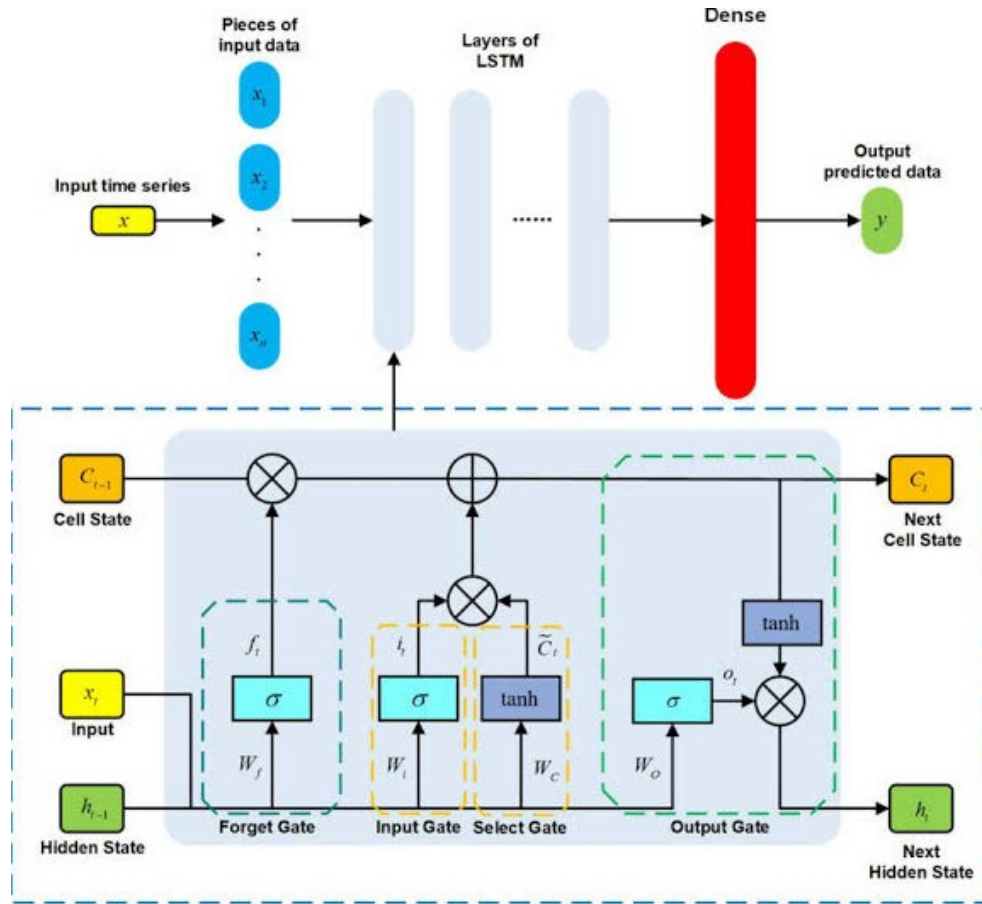


Figure 2: Architecture of LSTM model [27].

operations that control the input gate (i_t), output gate (o_t), and forget gates (f_t), which are managed in cell memory. ($\tilde{C}_t$) is the initial cell state, (C_t) is the updated cell state and (h_t) is the hidden value updated at every time step t . The filtered cell state is passed through the activation function which predicts what portion should appear as the output of current LSTM unit. The output (h_t) from current LSTM block is passed through the SoftMax layers to get the predicted output (y_t) from the current block.

After the LSTM layer processes the input sequence, an output vector V_t which contains raw values is obtained. According to Agarap [26], the SoftMax function is applied to partition the output such that the total sum is 1, which is equivalent to a categorical probability distribution. The final layer comprises a single neuron for each of the attack classes, and each attack class produces a value between 0 and 1, which is directly interpretable as probabilities.

The SoftMax function is applied elementwise to the vector V_t , for each V_i in V_t , the SoftMax:

$$(V_i) = \frac{e^{v_i}}{\sum_{n_i=1} e^{v_j}}, \quad (13)$$

where V_t = output vector of LSTM, V_i = element in V_t . The resulting values will be between 0 and 1, and they will sum up to 1.

3.5.4. Architecture of the RF-LSTM hybrid model

The RF and LSTM models were trained independently on the preprocessed data. The same input data is fed into both the trained RF and LSTM models. The RF model make prediction based on its learned decision rules, also the LSTM model makes prediction based on it learned temporal patterns. The final predictions of both RF and LSTM are combined using a meta-model. As shown in Figure 3, weight is assigned to each model according to its performance, and the outcome of the combination, is the final prediction of the new RF-LSTM hybrid model. By leveraging the strengths of both RF and LSTM, this meta-classifier optimizes their combination and offers a powerful and effective solution for intrusion detection of majority and minority class attacks simultaneously.

3.6. System evaluation

The effectiveness of the trained model was evaluated using confusion matrix and Matthews correlation coefficient (MCC). The testing set was used to evaluate the model's ability to detect intrusion accurately. Some hyperparameters like; (epoch, learning

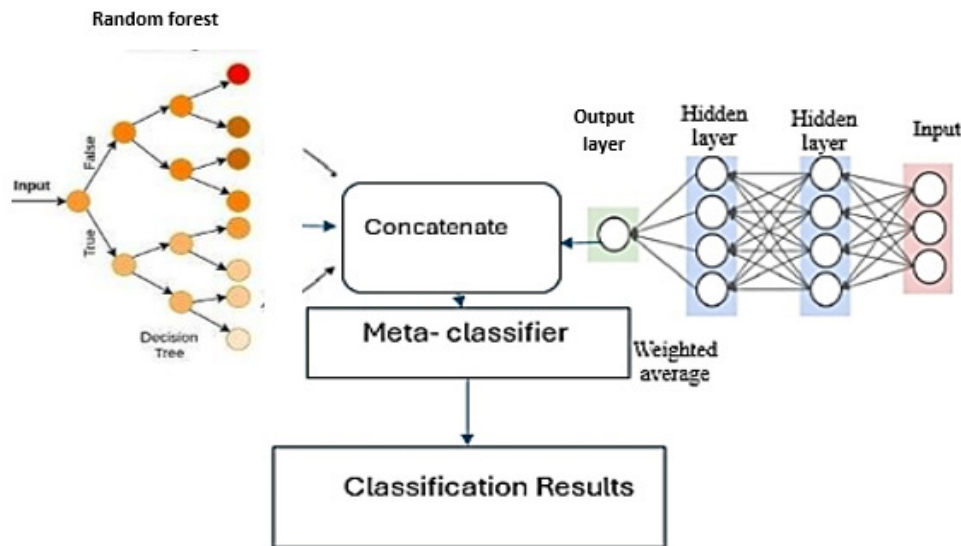


Figure 3: Architecture of the RF-LSTM.

rate) were adjusted to refine the model. The k -fold (where $k = 5$) cross validation method, which divides the dataset into two subsets, one for training and one for testing, was used to gauge the classifiers' performance. After performing the operation k times independently, the average of the k performances is determined and given back. This approach has the benefit of using the complete dataset for testing and training, thereby increasing the accuracy of the evaluation. Because some attacks, such as U2R and R2L, will decrease for each subset if we increase k , or they can be ignored during treatment, a fewer cross validation (only 5-fold) was used to assess the models.

NSL-KDD dataset was preferred due to its manageable size which ensures consistent evaluation and results comparison across different studies. While datasets like CICIDS2017 and UNSW-NB15 aim for realism, they contain a very huge number of normal traffic records compared to attack records resulting in class imbalance challenges. Comparatively, NSL-KDD dataset has more balanced distribution of normal and attack records since redundant records are removed. Consequently, minority class attacks like R2L and U2R are effectively detected by models trained by this dataset. Also, since NSL-KDD is a smaller non-redundant dataset, researchers can use all of it for training and testing their models. This makes sure that their experimental results are consistent and can be reproduced by others, hence providing a dependable benchmark for comparing several other intrusion detection systems (IDS) algorithms. In addition, having many existing research papers and already published results on the NSLKDD dataset makes it easy to compare a new model's performance directly against many existing techniques. This helps validate the new model's effectiveness and shows its contribution to the field.

4. Results and discussions

The approach adopted in obtaining the attack distribution across protocols was using the Machine Learning libraries which is the best approach for large dataset. The Pandas (Python libraries) was used to group the data by protocols and calculate attack type distribution. By analyzing the attack distribution across different protocols in the NSL KDD dataset, the researchers gained valuable insight into vulnerabilities associated with different communication protocols. Figure 4 and Figure 5 respectively, shows the overall attack volume across different protocols. It reveals which protocol is more susceptible to certain attack type. It is also seen that attacks like rootkit, guess password, buffer overflow, warezmaster, and warezcient (which all fall under the minority attack class), were sufficiently classified with a recall of about 90%.

4.1. Performance indicators

- True positive rate:

$$TPR = \frac{TP}{(TP + FN)}. \quad (14)$$

Correctly classified intrusion instances (e.g., DoS, U2R, R2L).

protocol_type	icmp	tcp	udp
attack			
back	0	956	0
buffer_overflow	0	30	0
ftp_write	0	8	0
guess_passwd	0	53	0
imap	0	11	0
ipsweep	3117	482	0
land	0	18	0
loadmodule	0	9	0
multihop	0	7	0
neptune	0	41214	0
nmap	981	265	247
normal	1309	53599	12434
perl	0	3	0
phf	0	4	0
pod	201	0	0
portsweep	5	2926	0
rootkit	0	7	3
satan	32	2184	1417
smurf	2646	0	0
spy	0	2	0
teardrop	0	0	892
warezclient	0	890	0
warezmaster	0	20	0

Figure 4: Protocol-types specific attack distribution.

- False positive rate:

$$FPR = \frac{FP}{(FP + TN)}. \quad (15)$$

Normal traffic instances incorrectly classified as intrusions.

- True negative rate:

$$TNR = \frac{TN}{(TN + FP)}. \quad (16)$$

Correctly classified normal traffic instances.

- False negative rate:

$$FNR = \frac{FN}{(FN + TP)}. \quad (17)$$

Intrusion instances incorrectly classified as normal traffic.

- Accuracy:

$$ACC = \frac{(TP + TN)}{(TP + TN + FP + FN)}. \quad (18)$$

- Detection rate:

$$DR = \frac{TP}{(TP + FN)}. \quad (19)$$

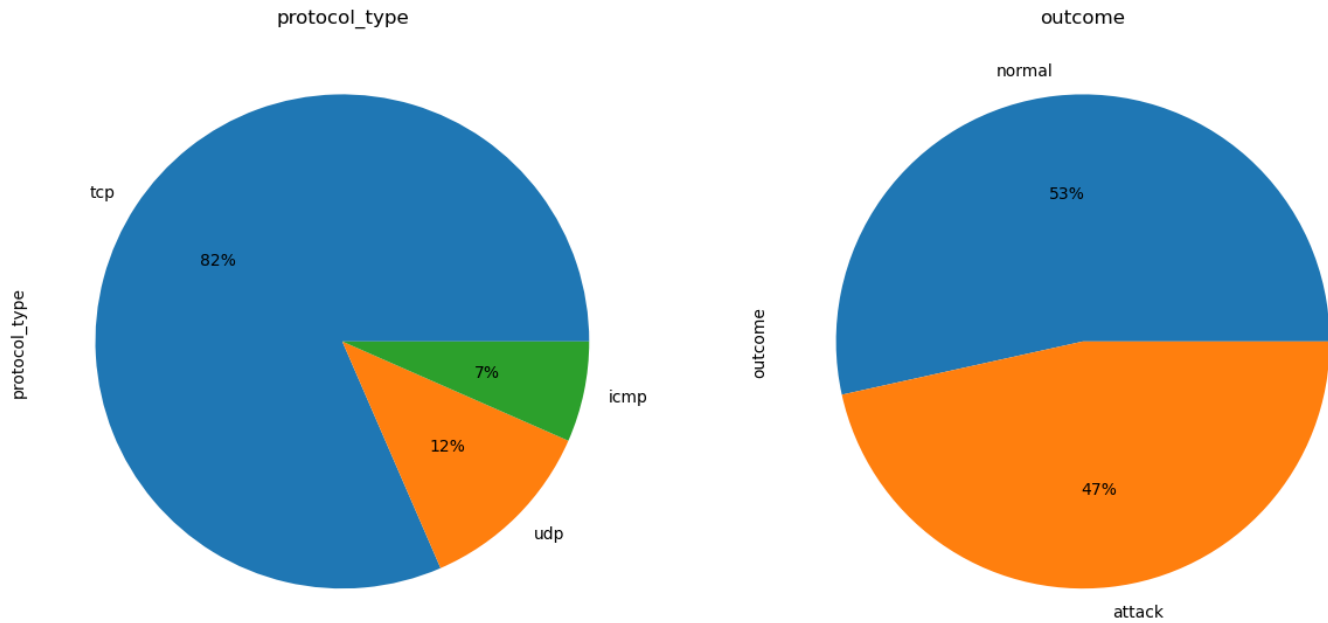


Figure 5: Attack distribution by protocol.

- False alarm rate:

$$FAR = \frac{FP}{(FP + TN)}. \quad (20)$$

- Precision:

$$P = \frac{TP}{(TP + FP)}. \quad (21)$$

- Recall:

$$R = \frac{TP}{(TP + FN)}. \quad (22)$$

- F1-score:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}, \quad (23)$$

where ACC is Accuracy, TP is True positive, FP is False positive, TN is True negative, FAR is False alarm rate, DR is Detection rate, FN is False negative, P is Precision, R is Recall.

4.2. Evaluation with LSTM

Figure 5 shows the performance value for the LSTM model as follows: Accuracy = $(TP + TN) / \text{Total instances} = (9338 + 6072) / 16875 = 15410 / 16875 = 0.9132 \approx 91\%$.

From a total of 9855 samples, 9338 were correctly classified as standard attacks, and out of 7459, 2421, 65, and 2743 samples, 6072, 1585, 16 and 2411 were correctly classified as DoS, Probe, U2R and R2L attacks respectively (Figure 6).

It can also be seen in Figure 7 that the hybrid model is a well performing model where its diagonal values (TN and TP) are high, whereas its off-diagonal values (FN and FP) are low. The performance accuracy of the new model is about 98.37%.

4.3. Simple algorithm for calculating the confidence interval

Original Test Data (100 samples)

Calculate Accuracy (98.3%)

Bootstrap Resampling (100 iterations)

Iteration 1: Sample with replacement → Calculate Accuracy₁

Iteration 2: Sample with replacement → Calculate Accuracy₂

...

Iteration 100: Sample with replacement → Calculate Accuracy₁₀₀

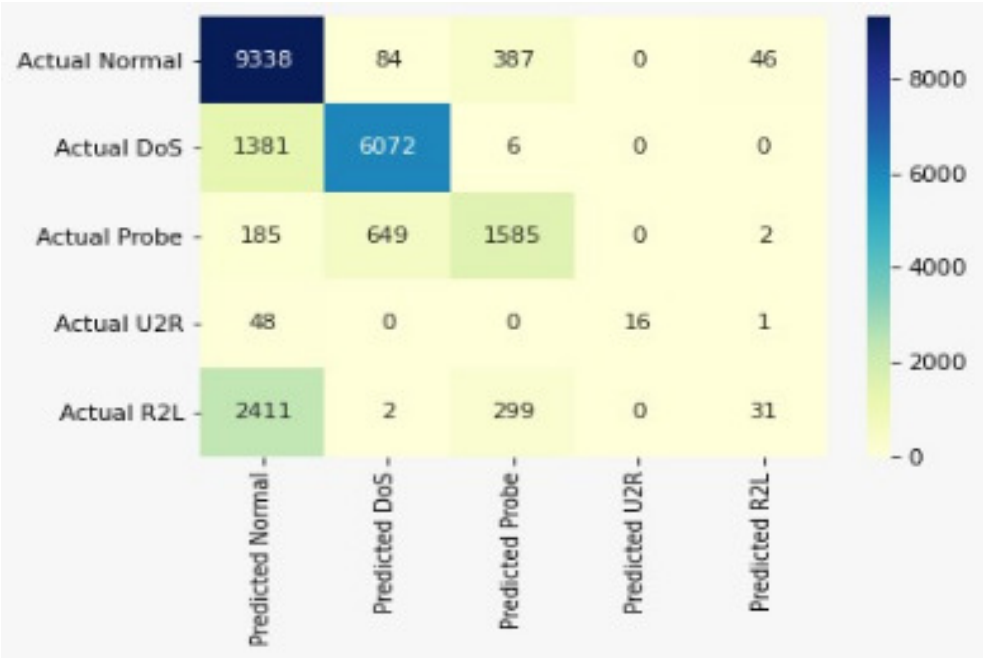


Figure 6: Evaluation with LSTM.

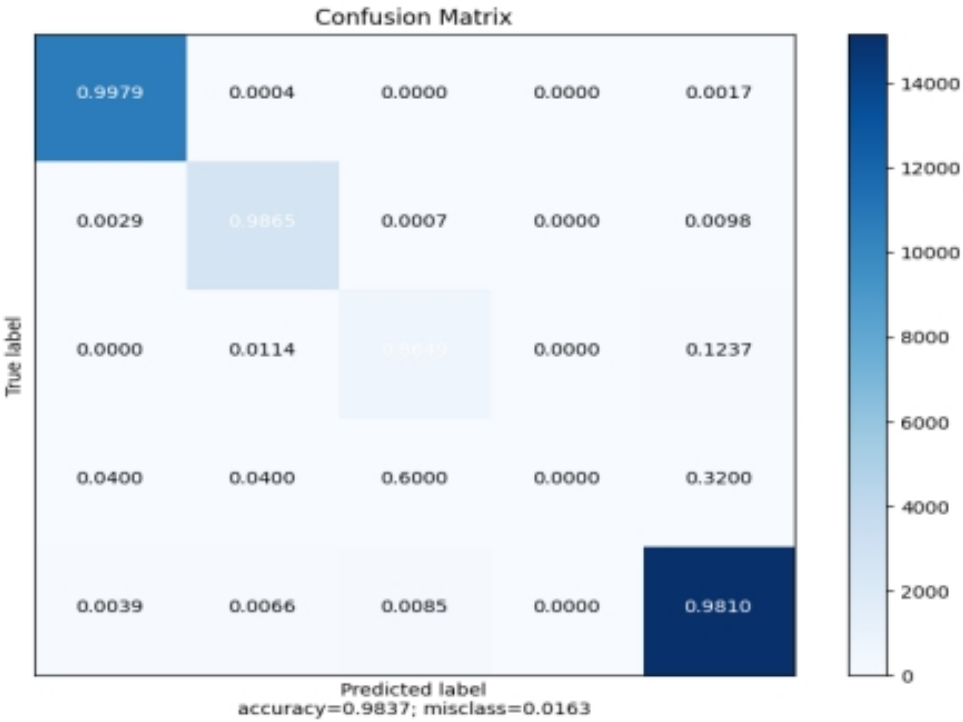


Figure 7: Performance accuracy for the RF-LSTM hybrid model.

Collection of 100 Bootstrap Accuracy Values
[96.2%, 97.1%, 95.8%, ..., 98.3%]
Sort Bootstrap Accuracy Values
(e.g., [95.8%, 96.0%, ..., 97.0%, ..., 98.0%, 98.9%])
Identify 2.5th Percentile (Lower Bound) and 97.5th Percentile (Upper Bound)
(e.g., 2.5th percentile = 96.0%, 97.5th percentile = 98.4%)
95% Confidence Interval: [Lower Bound, Upper Bound]

```

epoch 1/100
20/20 [=====] - 9s 139ms/step - loss: 1.5092 - accuracy: 0.7948 - val_loss: 1.3103 - val_accuracy: 0.8396
Epoch 2/100
20/20 [=====] - 1s 71ms/step - loss: 0.9293 - accuracy: 0.8389 - val_loss: 0.4878 - val_accuracy: 0.8416
Epoch 3/100
20/20 [=====] - 1s 69ms/step - loss: 0.4405 - accuracy: 0.8472 - val_loss: 0.3873 - val_accuracy: 0.8695
Epoch 4/100
20/20 [=====] - 1s 70ms/step - loss: 0.3519 - accuracy: 0.9024 - val_loss: 0.3070 - val_accuracy: 0.9250
Epoch 5/100
20/20 [=====] - 1s 69ms/step - loss: 0.2690 - accuracy: 0.9298 - val_loss: 0.2245 - val_accuracy: 0.9368
Epoch 6/100
20/20 [=====] - 1s 71ms/step - loss: 0.2061 - accuracy: 0.9389 - val_loss: 0.1843 - val_accuracy: 0.9438
Epoch 7/100
20/20 [=====] - 2s 102ms/step - loss: 0.1739 - accuracy: 0.9446 - val_loss: 0.1587 - val_accuracy: 0.9493
Epoch 8/100
20/20 [=====] - 1s 70ms/step - loss: 0.1509 - accuracy: 0.9501 - val_loss: 0.1375 - val_accuracy: 0.9532
Epoch 9/100
20/20 [=====] - 1s 70ms/step - loss: 0.1312 - accuracy: 0.9544 - val_loss: 0.1181 - val_accuracy: 0.9553
Epoch 10/100
20/20 [=====] - 1s 69ms/step - loss: 0.1125 - accuracy: 0.9571 - val_loss: 0.1009 - val_accuracy: 0.9607
Epoch 11/100
20/20 [=====] - 1s 69ms/step - loss: 0.0992 - accuracy: 0.9640 - val_loss: 0.0894 - val_accuracy: 0.9729
Epoch 12/100
20/20 [=====] - 1s 70ms/step - loss: 0.0895 - accuracy: 0.9700 - val_loss: 0.0816 - val_accuracy: 0.9740
Epoch 13/100
20/20 [=====] - 1s 70ms/step - loss: 0.0836 - accuracy: 0.9722 - val_loss: 0.0756 - val_accuracy: 0.9761
Epoch 14/100
20/20 [=====] - 1s 71ms/step - loss: 0.0793 - accuracy: 0.9739 - val_loss: 0.0713 - val_accuracy: 0.9772
Epoch 15/100
20/20 [=====] - 1s 70ms/step - loss: 0.0746 - accuracy: 0.9742 - val_loss: 0.0675 - val_accuracy: 0.9778
Epoch 16/100
20/20 [=====] - 1s 71ms/step - loss: 0.0718 - accuracy: 0.9758 - val_loss: 0.0638 - val_accuracy: 0.9801
Epoch 17/100
20/20 [=====] - 1s 70ms/step - loss: 0.0682 - accuracy: 0.9769 - val_loss: 0.0617 - val_accuracy: 0.9788
Epoch 18/100
20/20 [=====] - 1s 69ms/step - loss: 0.0658 - accuracy: 0.9779 - val_loss: 0.0585 - val_accuracy: 0.9820
Epoch 19/100
20/20 [=====] - 1s 69ms/step - loss: 0.0627 - accuracy: 0.9792 - val_loss: 0.0565 - val_accuracy: 0.9828
Epoch 20/100
20/20 [=====] - 1s 70ms/step - loss: 0.0619 - accuracy: 0.9789 - val_loss: 0.0550 - val_accuracy: 0.9821
Epoch 21/100
20/20 [=====] - 1s 70ms/step - loss: 0.0592 - accuracy: 0.9804 - val_loss: 0.0544 - val_accuracy: 0.9829
Epoch 22/100
20/20 [=====] - 1s 73ms/step - loss: 0.0593 - accuracy: 0.9798 - val_loss: 0.0528 - val_accuracy: 0.9826
Epoch 23/100
20/20 [=====] - 1s 70ms/step - loss: 0.0564 - accuracy: 0.9811 - val_loss: 0.0517 - val_accuracy: 0.9835
Epoch 24/100

```

Figure 8: Epoch for train and test dataset.

Table 2: Overall performance of the model for lr = 0.01.

Metrics Epochs	50	70	100
ACCURACY	97.92	98.19	98.37
FPR	2.06	1.81	1.62
PRECISION	96.86	96.94	96.98
RECALL	96.34	96.52	96.58
F1 SCORE	96.93	96.96	97.03

(e.g., [96.0%, 98.4%])

The confidence interval does not contain a null value, suggesting the result is statistically significant at the 95% confidence level.

4.4. Epochs

Multiple epochs were typically required for the model to learn effectively and capture the complex patterns that differentiate normal from intrusive traffic. Figure 8 shows the different epochs through the entire training dataset. The model learns and adjusts its internal parameters to improve its ability to distinguish between normal and malicious network traffic. A total of 100 epochs were considered, and during the course of repetitive learning, it is seen that the accuracy improves over time as the epoch increases as shown in Figure 8.

Figure 9 shows that accuracy increases over time as the epoch increases. This indicates better detection capability of the hybrid model. While Figure 10 shows the loss function steadily decreasing over epochs as the model learns and minimizes prediction errors. The decreasing loss indicates better model performance.

From Table 2, it is seen that all the metrics considered apart from False Positive Rate (FPR) are directly proportional to number of iteration (epoch). for example; all the values for accuracy, precision, recall and F1 score at 100 epoch, are greater than the respective values at 50 and 70 epochs.

To ascertain the stability of the hybrid model in making prediction, a Precision - Recall analysis was performed by plotting the precision rate (PR) against recall rate (RR) on the x and y axis respectively as seen in figure 11, which shows that the hybrid model has high stability and is very apt in making decisions.

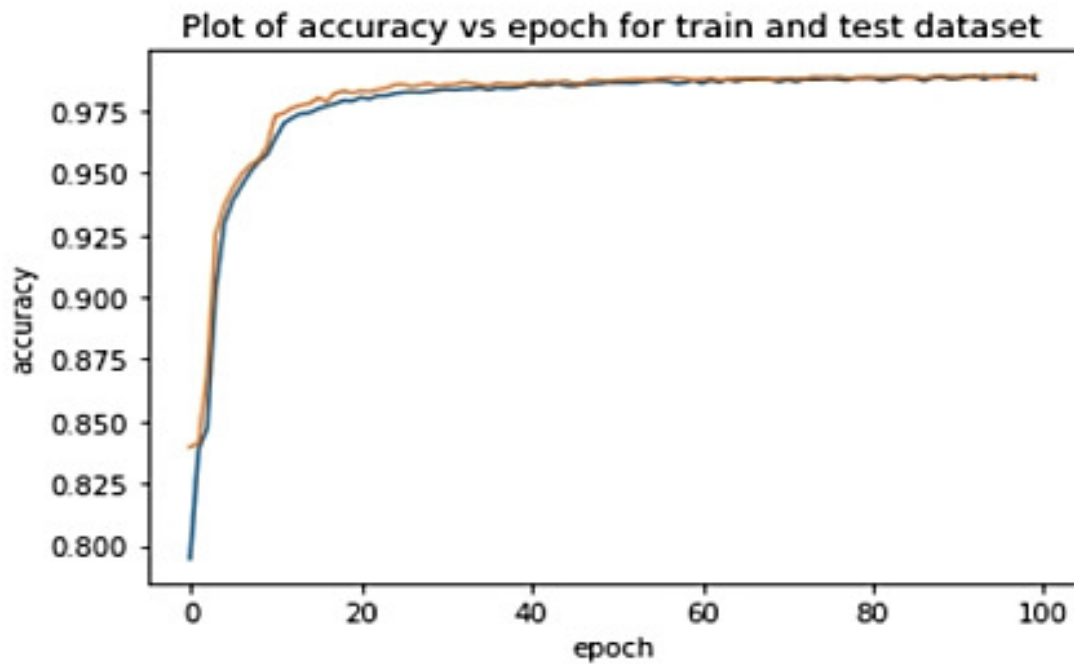


Figure 9: Accuracy vs epoch.

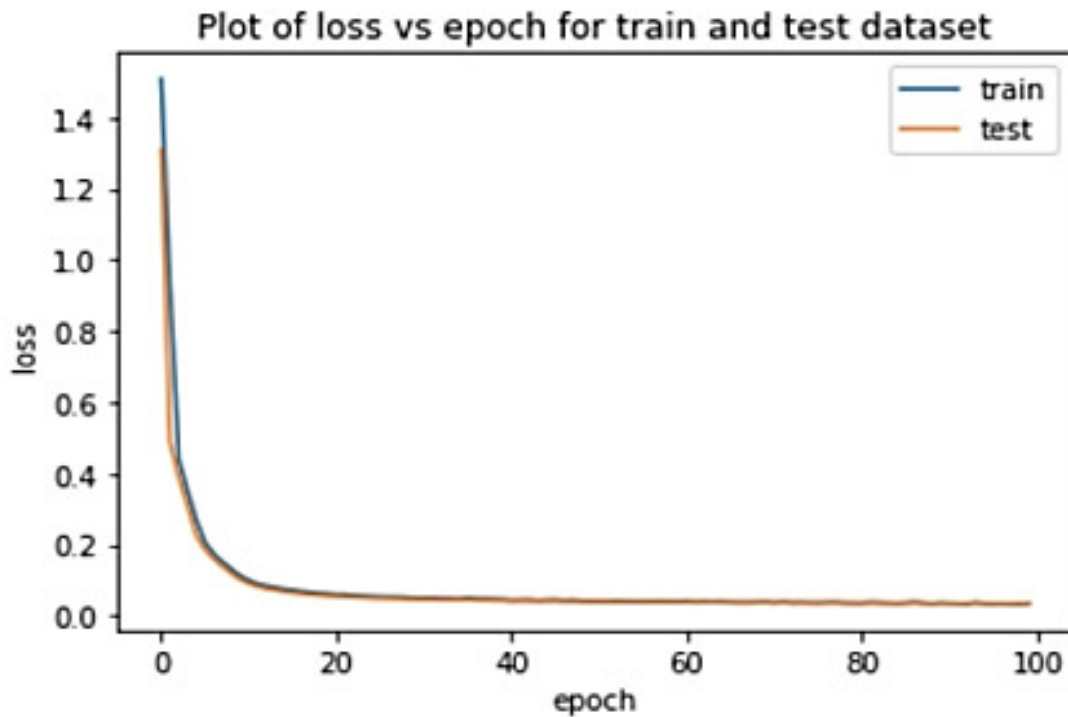


Figure 10: Loss vs epoch.

$$\text{Precision} = TP / (TP + FP) = 39800 / 41040 = 0.9698 \approx 96.98\%.$$

$$\text{Recall} = TP / (TP + FN) = 39828 / 41069 = 0.9698 \approx 96.98\%.$$

Also, Table 3 and Figure 12 respectively, shows the performance of the RF-LSTM model against some selected current state-of-the-art methods. While the RF-LSTM model could effectively classify both majority and minority class attacks simultaneously,

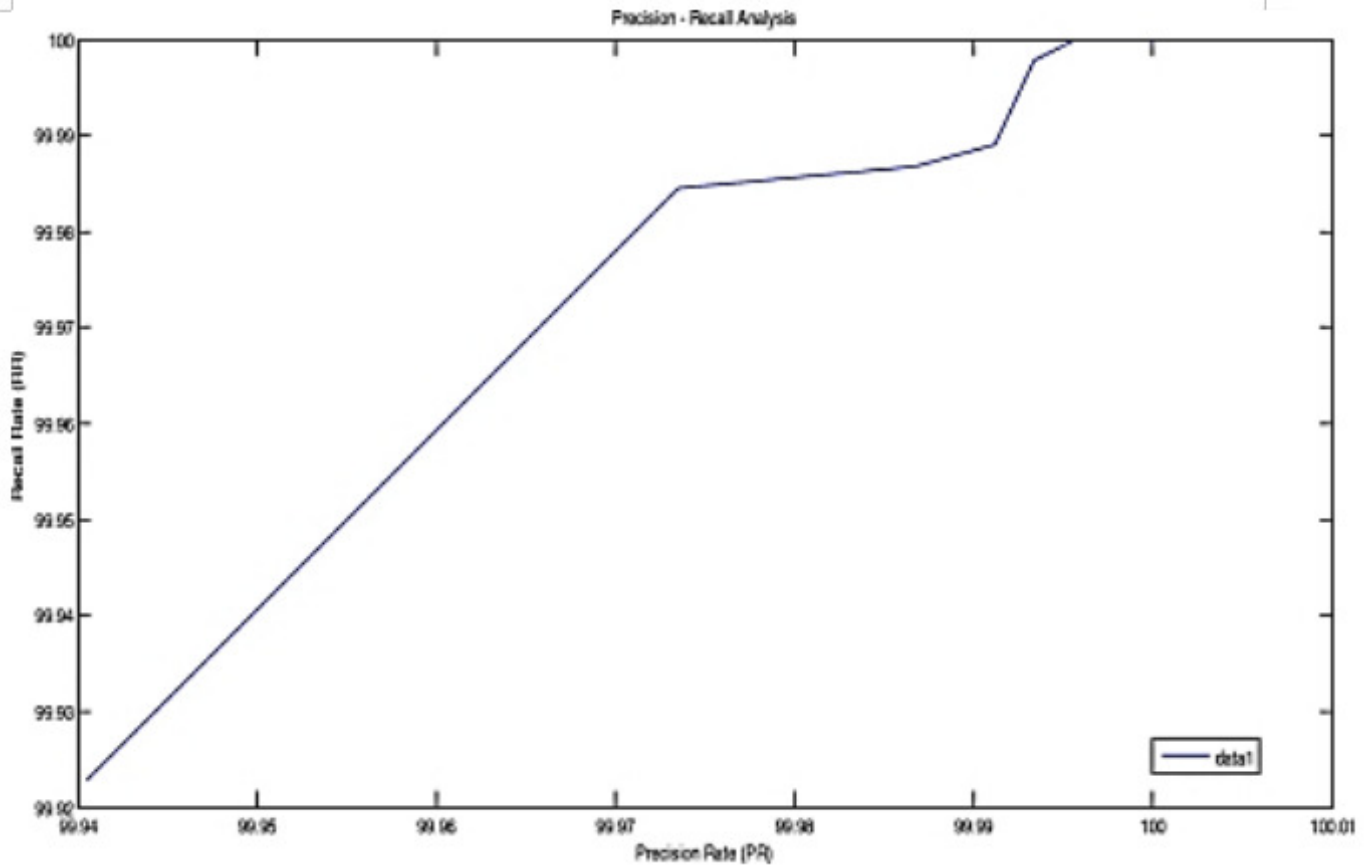


Figure 11: PR-RR graph.

Table 3: Comparing the performance of hybrid RF-LSTM model with current state-of-the-art methods.

Study	Accuracy	False Positive Rate	Impact on minority attack class
Current Study (Hybrid RF-LSTM model)	98.37%	1.62	90%
Prasanna <i>et al.</i> (2020) [9]	96.78%	3.22	Biased towards majority attack class
Devulapalli (2021) [6]	97.60%	2.40	50%
Meftah <i>et al.</i> (2019) [21]	86.04%	13.96	Biased towards majority attack class
Vinayakumar <i>et al.</i> (2019) [10]	93.5%	6.45	Biased towards majority attack class

Table 4: Comparison of the hybrid RF-LSTM model on different datasets.

Datasets	Accuracy	False Positive Rate
NSL-KDD	98.37%	1.62
UNSW-NB15	97.86%	2.14
CIC-IDS2017	98.18%	1.82
CTU-13	96.43%	3.56

models of the other approaches were biased towards the majority class attacks.

The RF-LSTM model was evaluated using other datasets as shown in Table 4. The minimum accuracy recorded was above 96%, the indication is that the hybrid RF-LSTM model is fit for purpose.

4.5. Model deployment

The hybrid model was deployed in a simulated network environment set up using virtual box, consisting of 3 systems and a host environment. The network was intruded via the host environment and the performance of the model was monitored and evaluated.

By hybridizing the strengths of RF and LSTM, the hybrid model on a practical network security scenario possesses the ability for early threat mitigation and incident response by flagging suspicious network activity as it happens. Networks with high value of

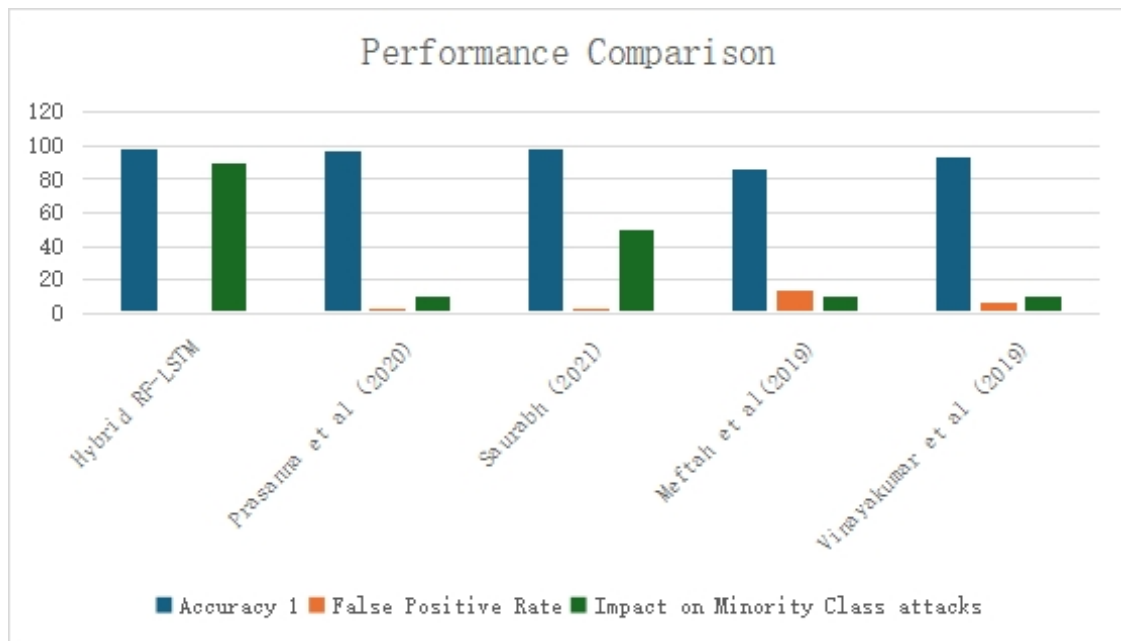


Figure 12: RF-LSTM vs other models.

time-series data will benefit much from this model's ability to detect critical anomalies in device communication pattern, examples being those on the Internet of things (IoT) or industrial control systems.

Though the hybrid RF-LSTM model is efficient in analyzing temporal features like packet size and frequency, it struggles to inspect payload. So future research could leverage the Network Traffic Analysis (NTA) Tools which provides visibility into both encrypted and unencrypted traffics. Also, decryption proxies could be deployed in controlled environment, but this process is complex to manage and raises significant privacy concerns.

5. Conclusion

The hybrid model was successfully implemented on NSL-KDD dataset. It proves to be very effective and reliable in detecting novel attacks and capturing minority class attacks. it also outperformed other models selected for comparison. Therefore, the research work is a promising approach for resolving major security issues. However, the proposed hybrid model while valuable for research and development, is far from being a perfect substitute for real-world testing. The real-world network is complex, dynamic and unpredictable, the simulated environment may fail to replicate the full diversity of real-world attacks. It is therefore recommended that future research study could utilize a hybrid data strategy by combining synthetic data with real-world data. The future models could be made more robust and resilient against sophisticated attacks by training them on adversarial data. This could be achieved by generating a subtle malicious sample that are carefully designed to dodge the model. This approach will enhance the ability of future models to generalize to new unseen threats. Future research could also use an ensemble of models instead of relying on a single hybrid RF-LSTM model to ensure network security is improved. An example could be combining a hybrid RF-LSTM with another single intrusion detection algorithms like Support Vector Machine (SVM), Convolutional Neural Network (CNN), Autoencoders etc.

Data availability

The complete Kaggle project, which includes the analytical notebooks and facilitates interactive exploration of the code and data, can be accessed from <https://www.kaggle.com/code/israelduff/ahena-bassey-nsf-kdd>.

Acknowledgment

The authors wish to express their gratitude to peer reviewers whose valuable feedback significantly improved both the clarity and rigor of this work.

References

- [1] A. Khraisat, I. Gondal, P. Vamplew & J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges", *Cybersecurity* **2** (2019) 20. <https://doi.org/10.1186/s42400-019-0038-7>.
- [2] J. Srinivas, A. K. Das & N. Kumar, "Government regulations in cyber security: framework, standards and recommendations", *Future Generation Computer Systems* **92** (2019) 178. <https://doi.org/10.1016/j.future.2018.09.0>.
- [3] A. Tesfahun & D. L. Bhaskari, "Intrusion detection using random forests classifier with SMOTE and feature reduction", in *International Conference on Cloud and Ubiquitous Computing and Emerging Technologies*, 2013. <https://doi.org/10.1109/CUBE.2013.31>.
- [4] T. T. Nguyen & V. J. Reddi, "Deep reinforcement learning for cyber security", *IEEE Transactions on Neural Networks and Learning Systems* **34** (2023) 3779. <https://doi.org/10.1109/TNNLS.2021.3121870>.
- [5] M. Ahmed, V. Deepak & G. S. Sajjan, "Comparative analysis of machine learning classifiers for network intrusion detection", in *Fourth International Congress on Information and Communication Technology*, 2020. <https://doi.org/10.1007/978-981-32-9343-4>.
- [6] S. Devulapalli, *A machine learning approach for uniform intrusion detection*, M.S. thesis, Purdue University Graduate School, Indiana, USA, 2021. <https://doi.org/10.25394/PGS.15032184.v1>.
- [7] N. Farnaaz & M. A. Jabbar, "Random forest modeling for network intrusion detection system", *Procedia Computer Science* **89** (2016) 213. <https://doi.org/10.1016/j.procs.2016.06.047>.
- [8] C. Yin, Y. Zhu, J. Fei & X. He, "A deep learning approach for intrusion detection using recurrent neural networks", *IEEE Access* **5** (2017) 21954. <https://doi.org/10.1109/ACCESS.2017.2762418>.
- [9] K. Prasanna, S. V. Sruthi, K. V. Kalyani & A. S. Tejaswi, "A CNN-LSTM model for intrusion detection system from high dimensional data", *Journal of Information and Computational Science* **10** (2020) 3. <https://doi.org/10.5281/zenodo.7911821>.
- [10] R. Vinayakumar, K. P. Soman & P. Poornachandran, "Applying convolutional neural network for network intrusion detection", in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, 1222. <https://doi.org/10.1109/ICACCI.2017.8126009>.
- [11] P. K. Bediako, *Long short-term memory recurrent neural network for detecting DDoS flooding attacks within tensorflow implementation framework*, M.S. dissertation, Department of Computer Science, University of Ghana, Accra, Ghana, 2017. <https://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-66389>.
- [12] A. Hammad, "Random forest and LSTM hybrid model for detecting DDoS attacks in healthcare IoT networks", *Cybersystems Journal* **1** (2024) 1. <https://doi.org/10.57238/csj.0kdtzj06>.
- [13] E. U. H. Qazi, M. H. Faheem & T. Zia, "HDLNIDS: Hybrid deep-learning-based network intrusion detection system", *Applied Sciences* **13** (2023) 4921. <https://doi.org/10.3390/app13084921>.
- [14] S. Bamber, A. V. Katkuri, S. Sharma & M. Angurala, "A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system", *Computers & Security* **134** (2024) 104146. <https://doi.org/10.1016/j.cose.2024.104146>.
- [15] F. Laghrissi, S. Douzi & K. Douzi, "Intrusion detection systems using long short-term memory (LSTM)", *Journal of Big Data* **8** (2021) 65. <https://doi.org/10.1186/s40537-021-00448-4>.
- [16] F. Omer, A. Hashim, F. Sali & A. Ahmad, "Binary classification of low-rate DoS attacks using long short-term memory feed-forward (LSTM-FF) intrusion detection system (IDS)", *Engineering Science and Technology, an International Journal* **66** (2025) 102049. <https://doi.org/10.1016/j.jestech.2025.102049>.
- [17] Y. Xue, C. Kang & H. Yu, "HAE-HRL: A network intrusion detection system utilizing a novel autoencoder and a hybrid enhanced LSTM-CNN-based residual network", *Computers & Security* **151** (2025) 104328. <https://doi.org/10.1016/j.cose.2025.104328>.
- [18] A. T. Azar, E. Shehab, A. M. Matter, I. A. Hameed & S. A. Elsaid, "Deep learning based hybrid intrusion detection systems to protect satellite networks", *Journal of Network and Systems Management* **31** (2023) 82. <https://doi.org/10.1007/s10922-023-09767-8>.
- [19] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi & R. Ahmad, "CNN-LSTM: Hybrid deep neural network for network intrusion detection system", *IEEE Access* **10** (2022) 99837. <https://doi.org/10.1109/ACCESS.2022.3206425>.
- [20] M. Sajid, K. R. Malik, A. Almogren, T. S. Malik, A. H. Khan, J. Tanveer & A. Rehman, "Enhancing intrusion detection: a hybrid machine and deep learning approach", *Journal of Cloud Computing* **13** (2024) 123. <https://doi.org/10.1186/s13677-024-00685-x>.
- [21] S. Meftah, T. Rachidi & N. Assem, "Network based intrusion detection using the UNSW-NB15 dataset", *International Journal of Computing and Digital Systems* **8** (2019) 478. <https://doi.org/10.12785/ijcds/080505>.
- [22] R. R. Devi & M. Abualkibash, "Intrusion detection system classification using different machine learning algorithms on KDD-99 and NSL-KDD datasets - A review paper", *International Journal of Computer Science and Information Technology* **11** (2019) 3. <https://doi.org/10.5121/ijcsit.2019.11306>.
- [23] M. Ozkan-Okay, R. Samet, O. Aslan & D. Gupta, "A comprehensive systematic literature review on intrusion detection systems", *IEEE Access* **9** (2021) 157727. <https://doi.org/10.1109/ACCESS.2021.3129336>.
- [24] Z. Ahmad, A. Shahid, W. Shiang, J. Abdullah & F. Ahmad, "Network intrusion detection system: a systematic study of machine learning and deep learning approaches", *Transactions on Emerging Telecommunications Technologies* **32** (2021) e4150. <https://doi.org/10.1002/ett.4150>.
- [25] Z. Wang, D. Huo, L. Huo & W. Yang, "An efficient network intrusion detection approach based on deep learning", *Wireless Networks* **27** (2021) 4967. <https://doi.org/10.1007/s11276-021-02698-9>.
- [26] A. F. Agarap, "Deep learning using rectified linear units (ReLU)", 2018. <https://doi.org/10.48550/arXiv.1803.08375>.
- [27] S. He, J. Liu, X. Zhu, Z. Dai & D. Li, "Research on modelling and predicting of BDS-3 satellite clock bias using the LSTM neural network model", *GPS Solution* **27** (2023) 108. <https://doi.org/10.1007/s10291-023-01451-3>.