Hindawi
Scientific Programming
Volume 2022, Article ID 6172362, 11 pages
https://doi.org/10.1155/2022/6172362

Research Article

Intrusion Detection for Cyber-Physical Security System Using Long Short-Term Memory Model

Gazi Md. Habibul Bashar,[1] Mohammod Abul Kashem,[2] and Liton Chandra Paul[3]

[1]*Institute of Information & Communication Technology, Dhaka University of Engineering & Technology, Gazipur, Bangladesh*
[2]*Department of Computer Science and Engineering, Dhaka University of Engineering & Technology, Gazipur, Bangladesh*
[3]*Department of Electrical, Electronic and Communication Engineering, Pabna University of Science and Technology, Pabna, Bangladesh*

Correspondence should be addressed to Liton Chandra Paul; litonpaulete@gmail.com

Received 25 July 2022; Revised 21 October 2022; Accepted 9 November 2022; Published 21 November 2022

Academic Editor: Jiangbo Qian

In the present context, the deep learning approach is highly applicable for identifying cyber-attacks on intrusion detection systems (IDS) in cyber-physical security systems. As a key part of network security defense, cyber-attacks can change and penetrate the security of the network system, then, the role of an IDS is to detect suspicious behaviors and act appropriately to protect the network from the onset of attacks. Machine learning and deep learning techniques are important for current intrusion detection systems. However, traditional intrusion detection systems are far from being able to quickly and accurately identify complex and diverse network attacks and obtained low accuracy and detection rates, thus, these methods frequently fail to manage big amounts of data in a vast network infrastructure and utilize a lot of features leads to poor performance. For addressing these issues and improving the accuracy and scalability, in this paper, we have implemented the deep learning method based on a new approach multilayer long short-term memory (LSTM) model for detecting attacks on a network. The novelty of the proposed scheme is that the optimum multilayer architecture is built to achieve maximum accuracy in the network architecture in order to boost performance using stacking multiple layers of LSTM cells in a more effective manner, and better stability to perform consistently in both binary classification and multiclass classification on NSL-KDD datasets. Experimental tests with KDDTest + datasets show that the proposed multilayer LSTM model provides outstanding results with 95% and 96% accuracy, respectively, in binary and multiclass classification. In order to deal with actual datasets and obtain good performance in the network design, our optimum multilayer architecture must be put into practice in order to execute real-time applications. Therefore, the results are better and more robust than the existing state-of-the-art methods.

1. Introduction

With the rapid growth of technologies and information, network security has become more significant over time as a result of the attention that businesses, industries, and enterprises have placed on systems such as cyber-physical security systems. In order to offer safe networks, cybersecurity experts discover the significance of creating an effective network intrusion detection system (IDS) [1, 2]. Cyber-physical systems (CPSs) embed sensing, computing, control, and networking into physical items and infrastructure, linking them to the Internet and to one another [3]. An IDS, a foundational layer, must quickly identify, assess, and react to harmful cyber traffic [4]. Network intrusion detection is important for monitoring and identifying potential threats [5]. There are often major imbalanced data in public datasets for intrusion detection. The management of large amounts of data in complex network infrastructure is another issue that these techniques typically fail to address [6]. For this reason, traditional IDSs based on conventional machine learning methods generally have some shortcomings, such as poor generalization ability, and low real-time performance. Over the last few decades, researchers have suggested a variety of intrusion detection systems employing machine learning, deep learning, and other statistical techniques [7]. In recent years, deep learning techniques have quickly developed and are now extensively used across many industries because of the constant expansion of massive data and computational capacity [8]. Both deep and conventional machine learning models were evaluated using well-known classification metrics [9]. In several fields, including image identification and natural language processing (NLP), deep learning has produced outstanding results. Numerous researchers have successfully used convolutional neural networks (CNN) to identify cyber-attacks in order to increase the intelligence and accuracy of network intrusion detection, but they have not yet made the anticipated breakthrough. The fact that network traffic is not in an image data format, is one of the primary causes of the failure. Given that network traffic data is often one-dimensional, one crucial approach processed the data using a recurrent neural network (RNN). Recently, an interesting attempt to use RNN-based models such as BiDLSTM, HLSTM, OCNN-HMLSTM, and Chi-square BidLSTM achieved low accuracy for feature selection resulting in weak performance. However, these methods' effectiveness significantly relies on simulated datasets. For the majority of classification models, these datasets are computationally costly since they frequently require numerous features for training. As a result, feature selection must be done prior to training in order to remove duplicate and unnecessary features from the datasets. For the majority of machine learning and deep learning models, feature selection is crucial at the data preprocessing stage. In this work, different processing steps were taken for numerical and category variables. A one-hot encoding procedure is used to translate the distinguishing characteristics into numerical values. Because we have carefully designed the model utilizing batch normalization and dropout layers in the right ways, we were able to avoid model overfitting. The characteristics with lower correlation have been eliminated. Since categorical features are handled via one-hot encoding, we have employed 87 features while taking into account the relevant dummy variables. In this paper, a multilayer LSTM$'$ network has been introduced. The multilayer LSTM offers not only the loop's end state but also includes all previous states as well. A dense representation for each time step was created, and the single LSTM layer may be given a sequential output. As a result, stacking numerous layers of LSTM cells more effectively and stabling them to work consistently is a considerable measure.

In this paper, we have developed a new approach called the optimum multilayer architecture of the LSTM model for intrusion detection systems in cyber-physical security systems in order to obtain greater accuracy, identify unknown intrusion attacks accurately, handle data processing, and classify imbalanced attack datasets, our new approach must be embedded for performing real-time applications. The key contributions of this research can be summarized as follows:

(i) This paper proposes the multilayer LSTM model that performs binary classification and multiclassification to evaluate the better detection performance metrics, enhances feature selection's efficiency, takes out lower correlation, effectively handles categorical features by one-hot encoding, and reduces error using the backpropagation algorithm.

(ii) The novelty of the proposed scheme, in which the optimum multilayer architecture is built to achieve maximum accuracy in the network architecture by stacking multiple layers of LSTM cells in a more effective manner, better stabling them to perform consistently, and leading to reduce model overfitting, and lead to enhance the overall performance using dropout layers and batch normalization properly.

(iii) Our optimum multilayer architecture has to be incorporated to run real-time applications with lower complexity because we have achieved decent performance on actual datasets in the network architecture, so real-world implementation of our model would give a better real performance and our approach outperforms existing state-of-the-art methods in the literature.

The remaining paper is organized as follows. Related works and methodology are described, respectively, in Section 2 and Section 3. The experimental results and discussion are presented in Section 4 and followed by a conclusion in Section 5.

2. Related Works

This section discusses the state-of-the-art in the use of deep learning for intrusion detection in the domain of cyber security. In recent years, deep learning-based intrusion detection in cyber-physical security systems has gradually become a popular research topic. Hou et al. [8], on the basis of HLSTM, an intrusion detection approach was developed that can learn across time levels on complicated network traffic sequences. On KDDTest+, multiclassification accuracy is 83.85%. However, they did not improve the ability to detect intrusion. In [10], the experiment showed that the accuracy of the RNN model is 81.29% for the test set KDDTest+ in the five-category classification. Ait Tchakoucht and Ezziyyani evaluated MLESM performance on multilabel classification, based on the hyperparameters as in binary and multiclass classification on NSL-KDD and KDD'99. But they obtained low accuracy for intrusion detection [11]. Shone et al. [12], proposed a novel classification model constructed from stacked NDAEs and the RF classification algorithm. They achieved an accuracy of 85.42% in multiclass classification. In [13], authors proposed SFSDT, which is a feature selection model for improving various RNN models in the IDS field. Imrana et al. [14], authors proposed chi-square BidLSTM, a novel IDS technique that combines a chi-square statistical model with a bidirectional long short-term memory (BidLSTM) model. According to the model's complexity and runtime evaluations, the suggested chi-square BidLSTM model is more complicated and requires more training time than the regular LSTM model. Kunang et al. [15] suggested the IDS model enhances the outcomes of attack categorization in intrusion detection by employing DL and a DAE for the pretraining process and fine-tuning using DNN via the HPO process. In [16], an efficient intrusion detection model had been developed by using the unified DL approach of OCNN-HMLSTM. The proposed OCNN extracts the spatial features while the HMLSTM extracts the temporal features and classifies the network data. Imrana et al. [17] proposed Chi-square bidirectional LSTM to identify network intrusions. The accuracy in multiclass is 95.62%. The created approach has a larger complexity and needs more training time than the typical LSTM model. These are the key limitations of BiDLSTM-based intrusion detection. However, all these works show that using a simple RNN layer as binary and multiclass classification does not yield significant performance metrics-accuracy, recall, F1-score, precision, and ROC-AUC score. In this work, we have used the multilayer LSTM model set to evaluate the performance in detecting network intrusions in both binary and multiclass classification on the NSL-KDD datasets and we have also compared it with BiDLSTM, HLSTM, OCNN-HMLSTM, Chi-square BidLSTM, and other deep learning methods proposed by previous researchers. Therefore, the proposed optimum multilayer architecture is well designed using batch normalization, dropout layers, and stacking multilayers of LSTM cells in a more appropriate approach in order to improve its stability and overall performance having the best results in comparison to the state-of-the-art methods in the literature.

3. Methodology

This section introduces the basic LSTM model, the proposed model architecture, and the proposed model methodologies based on the multilayer LSTM model in detail.

3.1. LSTM Model. Long short-term memory networks also known as "LSTM" are a variant of RNN which can learn long-term dependencies and are proposed by Hochreiter and Schmidhuber (1997). LSTMs were created to address the vanishing gradient issue for detecting anomalies in network traffic or IDSs and unsegmented data (intrusion detection systems) that might arise when regular RNNs are being trained [6]. The structure diagram of a single LSTM cell and the flowchart of the working procedure of the LSTM cell is depicted in Figures 1 and 2, respectively. The forget gate layer examines the input data and the data received from the previously hidden layer, then uses a sigmoid function to determine which information the LSTM will erase from the cell state. Cell state and forget gate can be calculated by the following equations:

$$C = f_t . c_{t-1} + i_t . \tilde{C}_t, \tag{1}$$

$$f_t = \left(W_f . \left[h_{t-1}, x_t\right] + b_f\right). \tag{2}$$

The input gate layer determines which data the LSTM will save in the cell state. A Tanh layer suggests a new vector to add to the cell state after choosing the input gate layer.

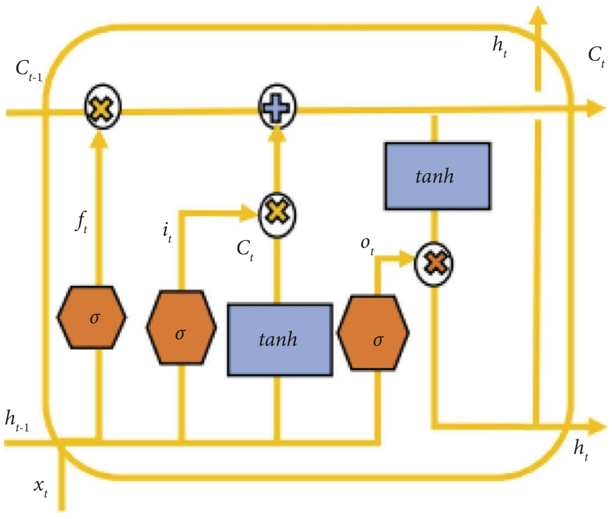


FIGURE 1: Structure diagram of a single LSTM cell.

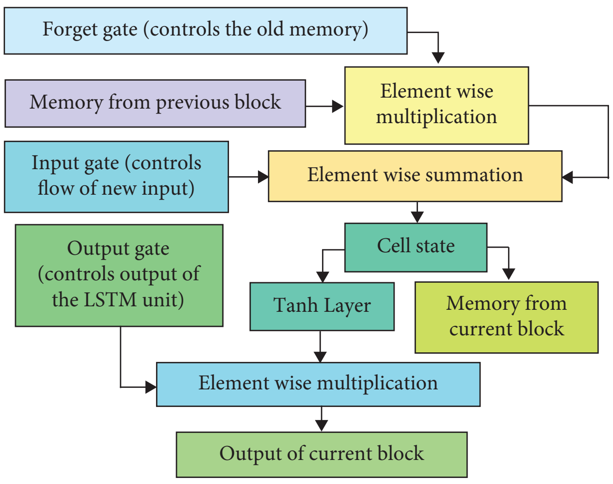

Forget gate (controls the old memory)
Memory from previous block
Element wise multiplication
Input gate (controls flow of new input)
Element wise summation
Output gate (controls output of the LSTM unit)
Cell state
Tanh Layer
Memory from current block
Element wise multiplication
Output of current block

FIGURE 2: Flowchart of the LSTM cell.

This information will be updated by using a sigmoid function. Then the LSTM updates the cell state by erasing the information. The input modulation gate (equation (3)) and input gate (equation (4)) can be calculated as follows:

$$\tilde{C}_t = \tan h\left(W_c . \left[h_{t-1}, x_t\right] + b_C\right), \tag{3}$$

$$i_t = \sigma\left(W_i . \left[h_{t-1}, x_t\right] + b_i\right). \tag{4}$$

In order to use this LSTM unit, we need to produce the output. Using the new memory, the prior output, and the current input, this step's output gate is managed. The result is sent via a Tanh layer to output just the information which we want to convey to the next neuron. The hidden state (equation (5)) and output gate (equation (6)) can be calculated as follows:

$$h_t = * \tan h\left(C_t\right), \tag{5}$$

$$O_t = \left(W_o . \left[h_{t-1}, x_t\right] + b_o\right), \tag{6}$$

where x_t is the current input, h_t and h_{t-1} are the hidden state of the t-th time and the $(t-1)$-th time, W_i, W_f, W_c, W_o, and B are the weight parameters in the self-recurrent, σ and tanh are sigmoid and hyperbolic tangent activation functions, respectively.

3.2. Proposed Model Architecture. LSTM is designed to tackle long-term dependency problems which is why it has a wide range of use cases. In addition, a single-layer LSTM is commonly utilized to transform sequences into dense, nonsequential features. These are the states at the conclusion of the RNN loop, and this phase is responsible for converting sequence data into tabular data.

3.2.1. Multilayer LSTM. Single-layer LSTM may not always be able to compress sequential data effectively enough. From our literature survey, we can see that sequential models are widely used to develop intrusion detection systems. Hence we have developed our model considering the LSTM cell as the basic building block. In the Keras framework deep learning toolkit, we have introduced a "multilayer LSTM" network, and we have initialized "n" LSTM cells. The multilayer LSTM, an evolution of this concept, features several hidden LSTM layers with different numbers of memory cells. As a result, each LSTM layer's hidden and cell states need to be initialized. The input vectors have been sent to the LSTM layer. The hidden state of the LSTM cell will be fed to the next layer. There are 4 levels of the LSTM layer each consisting of 100 cells. Different cells in the same layer are interconnected, and even different cells, as well as different layers, are also interconnected. Hence, they are sharing information between layers. This is making a powerful feature structure and allows us to explore the latent relationship. An input is necessary for each LSTM memory cell. Each memory cell in an LSTM will produce a single output for the whole input sequence of time steps while processing a single input sequence of time steps. It is seen in the model below, which features a single hidden LSTM layer that serves as both the input and output layers. If the preceding LSTM layer generates an output that serves as the input for the following layer, we can keep adding hidden LSTM layers.

In the network architecture, when we were adding more layers and cells caused overfitting and removing any of the layers which declined the performance. A dropout layer is imposed in each fully connected layer to reduce the overfitting in the system. Thus, we have properly designed it, and as result, stacking multiple layers of LSTM cells in a more effective manner and stabling them to perform consistently is a great deal. The proposed optimum multilayer architecture of the LSTM model for IDS is shown in Figure 3.

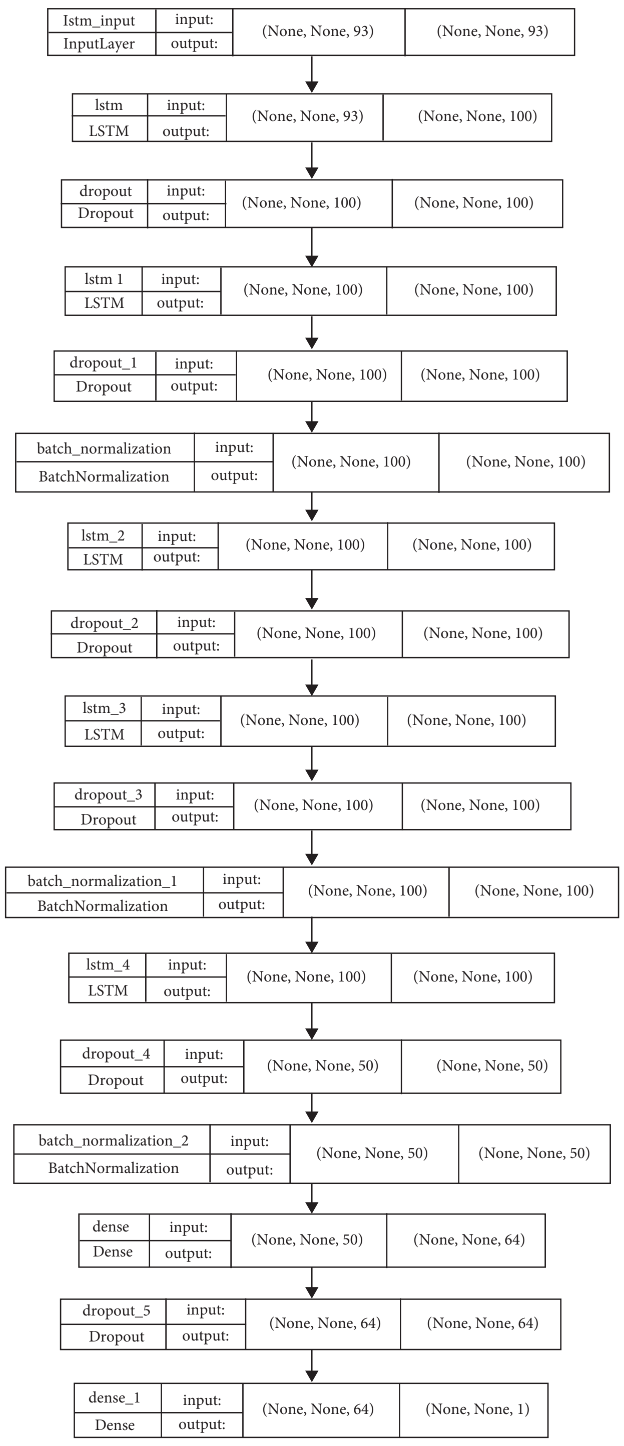

lstm_input | InputLayer | input: (None, None, 93) | output: (None, None, 93)
lstm | LSTM | input: (None, None, 93) | output: (None, None, 100)
dropout | Dropout | input: (None, None, 100) | output: (None, None, 100)
lstm 1 | LSTM | input: (None, None, 100) | output: (None, None, 100)
dropout_1 | Dropout | input: (None, None, 100) | output: (None, None, 100)
batch_normalization | BatchNormalization | input: (None, None, 100) | output: (None, None, 100)
lstm_2 | LSTM | input: (None, None, 100) | output: (None, None, 100)
dropout_2 | Dropout | input: (None, None, 100) | output: (None, None, 100)
lstm_3 | LSTM | input: (None, None, 100) | output: (None, None, 100)
dropout_3 | Dropout | input: (None, None, 100) | output: (None, None, 100)
batch_normalization_1 | BatchNormalization | input: (None, None, 100) | output: (None, None, 100)
lstm_4 | LSTM | input: (None, None, 100) | output: (None, None, 100)
dropout_4 | Dropout | input: (None, None, 50) | output: (None, None, 50)
batch_normalization_2 | BatchNormalization | input: (None, None, 50) | output: (None, None, 50)
dense | Dense | input: (None, None, 50) | output: (None, None, 64)
dropout_5 | Dropout | input: (None, None, 64) | output: (None, None, 64)
dense_1 | Dense | input: (None, None, 64) | output: (None, None, 1)

FIGURE 3: Proposed optimum multilayer architecture of the LSTM model.

3.2.2. Dropout Layers. Between the last hidden layer and the output layer, as well as between the first two hidden layers, it is applied. Input and recurrent connections to LSTM units are probabilistically eliminated from activation and weight updates during network training using dropout. Dropout layers are introduced in between each layer. It operates by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 during each update of the training phase. We have used a dropout layer to reduce overfitting leading to enhancing the performance of the model. Moreover, we have employed a dropout value of 0.2 which indicates that the units in that layer have a 20% chance of remaining active and an 80% chance of being dropped.

3.2.3. Batch Normalization. In order to speed up the training of multilayer LSTM model networks, batch normalization attempts to minimize internal covariate shifts. This is done by fixing the means and variances of the input layers during a normalization stage. At each hidden layer, batch normalization is the hidden weapon that addresses the unstable gradient issue for many of the deep learning architectures [18]. This eliminates the possibility of divergence and enables the use of considerably greater learning rates. This model is regularized using batch normalization, which lessens the requirement for dropout. We have also used batch normalization layers to speed up the training and handle internal covariate shifts thus leading to reduce overfitting. The batch normalization layer first determines the mean (μ) and the variance (σ^2) of the activation values across the batch, using the following equations:

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i, \tag{7}$$

$$\sigma_\beta^2 = \frac{1}{m}\sum_{i=1}^{m}\left(x_i - \mu_\beta\right)^2. \tag{8}$$

The batch normalization approach helps to speed up network convergence while maintaining the neural network's capacity for representation, whereas $B = \{x_{1\ldots m}\}$ represents the activation value inside a batch, α, β represents the parameters and $\hat{x}_i$ represents the value after normalization and y_i indicates the value after the batch normalization transformation. $\hat{x}_i$, and y_i can be estimated by using the following equations:

$$\hat{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 - \epsilon}}, \tag{9}$$

$$y_i = \alpha * \hat{x}_i + \beta. \tag{10}$$

3.2.4. Dense Layer. A dense layer is a neural network where all of the previous layer's neurons are connected to the inner neurons. The neurons in each layer of the neural network calculate the weighted average of the input, and the weighted average is then passed through a nonlinear function known as the "activation function." Finally, we have implemented the last LSTM layer which has 50 cells followed by a dropout layer with dropout chances of 0.2 and a batch normalization layer. There is another dropout layer with a dropout probability of 0.5. The final output of the layer then feeds into a dense layer of 64 neurons having ReLU as an activation function.

3.2.5. Rectified Linear Unit (ReLU). The most popular activation function of the deep learning model is the rectified linear unit. In essence, the function returns 0 if it receives a negative input, and if it receives a positive value, the function will return back the same positive value. We have trained the function that has addressed the drawbacks of several previous activation functions.

3.2.6. Cross Entropy Function. We have trained a classification model to categorize data by estimating the likelihood that the data belongs to one class or another, the cross entropy loss function is employed as an optimization function. For the binary classification task, the last layer consists of 1 neuron with a sigmoid function. For multiclass classification tasks, the last layer consists of 5 neurons with categorical cross entropy function. The formula for binary cross entropy (equation (11)) and multiclass or categorical cross entropy (equation (12)) can be presented as follows:

$$\mathrm{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i.\log\left(\hat{y}_i\right) + \left(1 - y_i\right).\log\left(1 - \hat{y}_i\right)\right), \tag{11}$$

$$\mathrm{CCE} = -\frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{J}\left(y_j.\log\left(\hat{y}_j\right) + \left(1 - y_j\right).\log\left(1 - \hat{y}_j\right)\right). \tag{12}$$

3.2.7. Adam Optimizer. The "exponentially weighted average" of the gradients is taken into account by this approach in order to speed up the gradient descent procedure. Our model is trained with Adam Optimizer with a learning rate of 0.0001 for 500 epochs. The Adam optimizer abides by the following equations:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{13}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{14}$$

3.2.8. Backpropagation Algorithm. This algorithm checks errors by going backward from output nodes to input nodes. It helps to evaluate the effect of a certain input variable on a network output. For the whole network, we have trained with a backpropagation algorithm. For backpropagation, we have created a list of mistakes and used visualization to see how the change in training affects the error. The mistake will be subtracted from the training data to gather and display accuracy. We have increased the overall performance metrics of the proposed model by reducing errors. Our training methodology is operating quite successfully.

In order to create our model, we used a Jupyter notebook and leveraged machine and deep learning libraries. We have thoroughly examined NSL-KDD datasets. When we have experienced this dataset and taken fewer layers in the network model, the result becomes low, and if we take more, the performance would be overfitting. Thus, the network architecture used batch normalization, dropout layers, and stacking many layers of LSTM cells in a more useful way. In order to improve their stability, maximum accuracy is obtained and optimum multilayer architecture is also explored, so real-world implementation of our optimum architecture would show a better real performance. Therefore, we can conclude that the proposed optimum multilayer architecture shown in Figure 3 would be workable in real applications.

3.3. Proposed Methodology. For the purpose of detecting network intrusion, the paper employed a multilayer LSTM model in which training is done using the NSL-KDDTest + datasets. As depicted in Figure 4, this process involves multiple steps.

3.3.1. Dataset. In this work, the NSL-KDDTest+ datasets are used for the detection of intrusion in a network (which is a newer version of the KDD99 CUP dataset). It is publicly available for use in the UNB data repository. It contains normal records as well as records of attacks such as denial of service (DoS) attacks that disrupt service availability, probe attacks that extract detailed information from servers, user-to-root (U2R) attacks that attempt to exploit system vulnerabilities in order to gain super user privileges, and remote to local (R2L) attacks that send packets to a machine over a network that has no account in order to lead to vulnerability issues and gain access to secure information [19]. The number of total samples of normal types and attack types was 22544 on NSL-KDDTest+.

3.3.2. Remove Null Values and Duplicate Rows. In the data set, there are lots of null values and it does not have a column name. Therefore, we have to provide a name and remove the attribute "difficulty_level," set descriptive statistics of the dataset as well as find a number of attack labels. Datasets are being changed attack labels to their respective attack class and calling the change_label() function and distribution of attack classes.

3.3.3. Standard Scaler or Normalization of Numerical Variables. In this step, the required libraries are imported and selected numeric attribute columns from the data. Then a standard scaler is applied to normalize and call the normalization() function.

3.3.4. One-Hot Encoding of Categorical Variables. A required library has been imported to plot and explores the distribution of normal and abnormal labels by using one-hot encoding. For multiclass classification, a data frame with multiclass labels such as Dos, Probe, R2L, U2R, and normal has been created where the label encoding 0, 1, 2, 3, and 4 are used for Dos, normal, probe, R2L, and U2R, respectively.

3.3.5. Feature Selection Based on Correlation. This work proposes a new feature selection algorithm to address the problem of data feature redundancy. Firstly, the algorithm calculates the importance of sample features by the random forest algorithm and ranks them in order of importance; then analyzes the correlation between features by Pearson's index, and finally combines the two results to select the

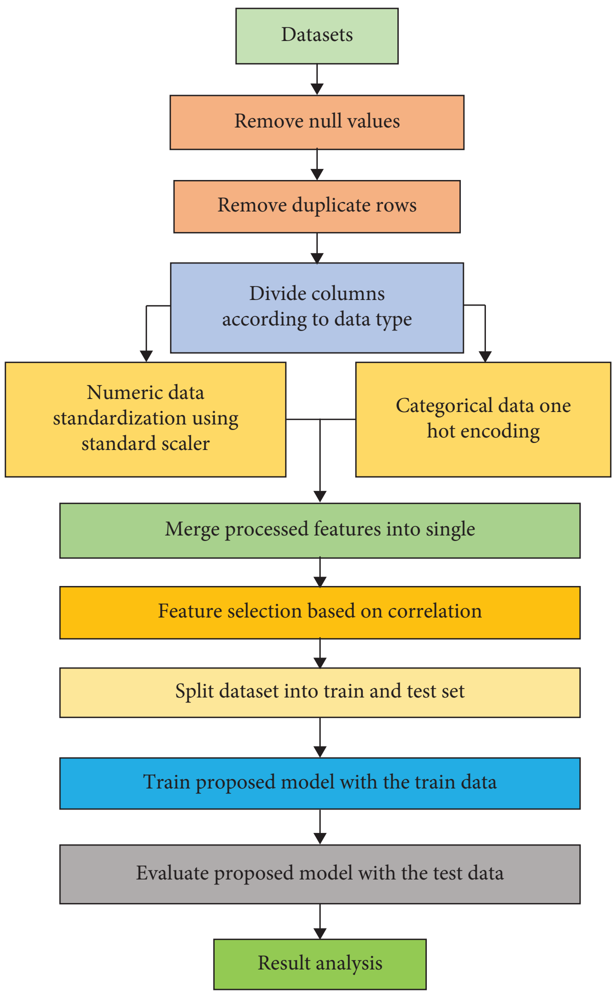

Datasets
Remove null values
Remove duplicate rows
Divide columns according to data type
Numeric data standardization using standard scaler
Categorical data one hot encoding
Merge processed features into single
Feature selection based on correlation
Split dataset into train and test set
Train proposed model with the train data
Evaluate proposed model with the test data
Result analysis

Figure 4: Methodology of the proposed model.

features. The pseudo-code of the feature selection algorithm is proposed as shown in the following algorithm:

3.3.6. Model Evaluation. To provide a comprehensive analysis of the proposed model, five evaluation metrics are used, including accuracy, precision, recall, F1-measure, and ROC-score. These metrics are calculated by following equations:

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}}, \quad (15)$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}, \quad (16)$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}, \quad (17)$$

$$F1 - \text{Measure} = 2\frac{\text{Precision } x \text{ Recall}}{\text{Precision} + \text{Recall}}, \quad (18)$$

A receiver operating characteristic (ROC) curve provides an overview of the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds for performance metrics.

```
    Input: data frame
    Output: selected feature
    Procedure:
(1) Creating a data frame with only numeric attributes of binary class and multiclass
(2) Encoded label attribute
(3) Finding the attributes which have more than 0.5 correlation with the encoded attack label attribute
(4) Selecting attributes found by using the Pearson correlation coefficient
(5) Joining the selected attribute with the one-hot encoded categorical data frame
(6) Joining one-hot encoded, and original attack label attribute
(7) Obtain the selected feature
```

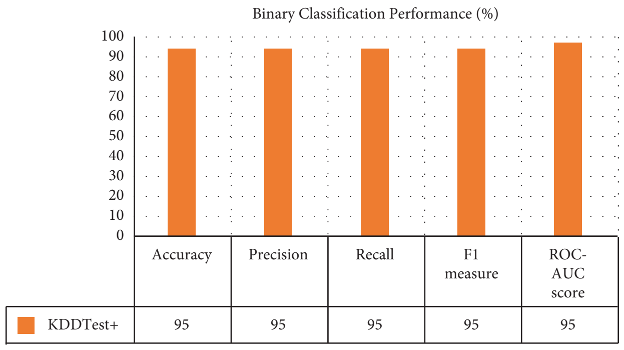
Algorithm 1: Feature selection algorithm (RFP).

4. Result and Discussion

This section is decorated with the results from different experiments and we have also provided a comparative analysis with some existing approaches.

4.1. Model Training and Validation. We have properly examined NSL-KDD datasets that are publicly available for use in the UNB data repository. All the experiments in this paper are performed using the Keras framework with the backend TensorFlow. These tests were carried out on a computer running Python 3.7 and Windows 10 with an Intel Corei7 CPU operating at 2.4 GHz, 8 GB of RAM, and 1 TB of secondary storage. In order to create our models, we used Jupyter notebook and leveraged machine and deep learning libraries. For binary classification or multiclass classification training, evaluation time depends on system parameters and specifications because deep learning requires good configuration for the fast execution of tasks. We have divided our data into a training set and a validation set in order to determine whether our model is an optimal fit. The validation set is primarily used to assess the model's performance; the training set is utilized to train the model. In our dataset, 80% of the data are used for model training, while the remaining 20% are taken to create the validation set of data. Training loss is an error in the training set of data. We encounter various errors after providing the trained network for the validation set of data. This error is known as a validation loss. The link between training loss and validation loss, as well as between training accuracy and validation accuracy, is explored below.

4.2. Results of Binary and Multiclass Classification. On the NSL-KDD dataset, we have applied both binary classification and multiclass classification experiments to assess the performance of the proposed model. This dataset is a completely balanced dataset, meaning for each target class there is an equal number of labels. We have eliminated duplicate features from the original dataset. After careful consideration, the accuracy rate can be a quite reliable performance metric for evaluating learning models. In the experiments, we have analyzed using evaluation metrics including precision, recall, F1-score, and accuracy for test sets. According to Table 1, it can be observed that the proposed optimum



	Accuracy	Precision	Recall	F1 measure	ROC-AUC score
🟧 KDDTest+	95	95	95	95	95

Figure 5: Performance of the proposed model for binary classification.

multilayer architecture exhibits the best performance compared to some cited approaches.

4.2.1. Binary Classification. In binary classification, we have carried out comprehensive experiments to evaluate the performance of the LSTM model on the NSL-KDDTest + dataset. There are two classes in the binary classification task, normal and attack/intrusion. Our proposed model achieves a high accuracy of 95%. The prime reason is that we have utilized network traffic data to learn across stacking multiple layers of LSTM cells in a more effective manner. By carefully reviewing the connection data of the targeted item, it is simple to identify such an attack and can be recognized quickly. As a result, we have obtained a decent classification outcome as shown in Figure 5. In the case of precision, a good number of positive class predictions have been obtained. Hence, the proposed approach performs more accurately (95%). For the F1-score, we are able to identify real threats correctly which is why false alarms do not disturb us, thus, we can achieve the highest values of 95% compared to other models by reducing error with the help of the backpropagation algorithm. For recall, more positive samples have been detected, therefore, our model obtained the best results and peaked at 95%. Furthermore, we achieved 98% of the ROC-AUC score which shows useful performance metrics that lead to a raised performance by employing dropout layers and batch normalizing. We have processed numerical and categorical features differently. This feature selection has been done prior to training in order to remove duplicate and unnecessary features from the datasets. We have discarded the features that have less correlation.

For binary classification, training accuracy and validation accuracy, and training loss and validation loss of the proposed model are illustrated in Figures 6(a) and 6(b), respectively. By evaluating the model's error on the training set, it measures the model's error. After every batch, we calculated the training loss. This is usually visualized by plotting a curve of the training loss. After each epoch, the validation loss is calculated. This helps us determine whether the model still needs to be adjusted. Both the training loss and the validation loss start to decline and then stabilize at a certain point. This denotes a model that is properly fitted, meaning it does not overfit or underfit. With increasing epochs, train error and validation error decrease. Train accuracy and validation accuracy rise quickly as train loss and validation loss decrease. It indicated that the proposed model is capable to avoid model overfitting for enhancing the overall model performance using stacking multiple layers of LSTM cells in a more effective strategy.

4.2.2. Multiclass Classification. We have also carried out a multiclass classification experiment on the same NSL-KDDTest + dataset to assess the effectiveness of the proposed multilayer LSTM model. There are 5 different classes for multiclass classification tasks including normal, DoS attack, R2L attack, Probe attack, and U2R attack. Due to one-hot encoding to handle categorical features, there are 87 features we have used considering corresponding dummy variables. The characteristic features are converted into numerical values using a one-hot encoding process. The results of the suggested multilayer LSTM model on the NSL-KDDTest + datasets are obtained with greater accuracy of 96% by applying effective feature selection. In this experiment, we have concentrated on a particular ReLU activation variant because, in earlier studies, this activation function in hidden layers produces the best performance. Figure 7 demonstrates outstanding performance for the multiclass dataset. The precision of 83% demonstrates the proposed model predicts positive. For the F1 score, we have accurately recognized serious dangers and are unperturbed by false alarms. Our model reaches the highest results with 93% for the F1-measure compared to some other cited works. Our model also shows the best outcomes (100%) for recall which is the percentage of a certain class correctly classified as positive to the total number of positive samples. Furthermore, we have got 98% of the ROC-AUC score, demonstrating a relevant performance metric.

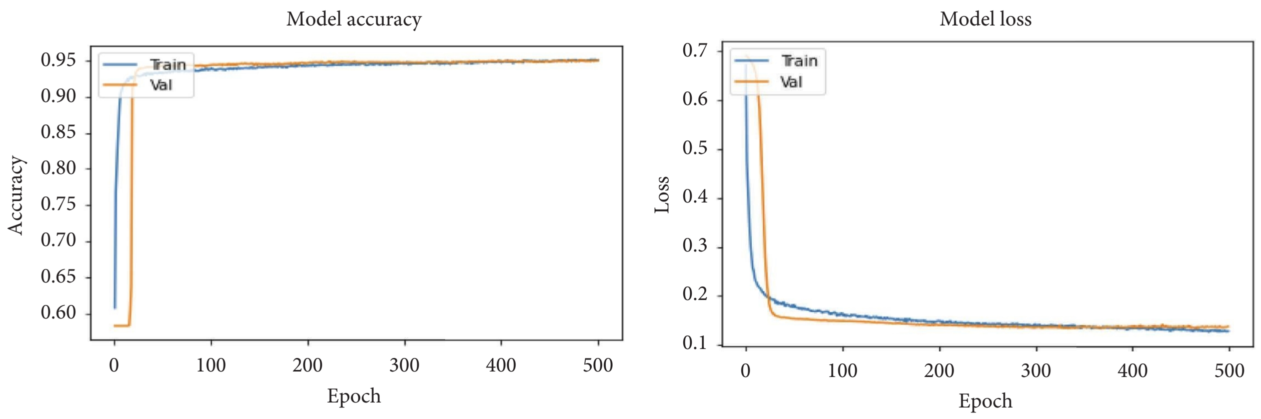
For multiclass classification, Figures 8(a) and 8(b) show training accuracy and validation accuracy, and training loss and validation loss of the suggested approach, respectively. At a certain point, the validation loss and training loss both start to decline and then stabilize. This suggests the model is well-fitted, meaning it is neither overfitting nor underfitting. These figures show that train error and validation error decrease as the number of epochs rises. Additionally, when train loss and validation loss fall, train accuracy and validation accuracy swiftly increase. It demonstrated that the proposed model obtained the optimal fit, which improved the model's overall performance.

4.3. Discussion. Comparing multiple algorithms in such a way is for reference purposes only. It is challenging for a model to function successfully in every setting since intrusion detection systems differ in their classification results.

4.3.1. For Binary Classification. Our proposed model shows the best performance compared to all models cited in Table 1 with an overall accuracy of 95% and an F-1 score of 95% in binary classification on the NSL-KDDTest + dataset and a ROC-AUC score of 98%. Yin et al. [10], who used an RNN-IDS, achieved only 83.28% accuracy for the KDDTest + dataset. Ait Tchakoucht and Ezziyyani [11] developed recurrent neural networks (RNN) with the multi-layered echo-state machine (ML-ESM) to model intrusion detection with a low accuracy of 83%. Imrana et al. [14] used a BiDLSTM method with a recall rate and accuracy of 90.79% and 94.26%, respectively. Rajesh Kanna and Santhi [16], the authors used OCNN-HMLSTM methods, achieving precision rate and accuracy of 86.71% and 90.67%, respectively. For the F1-score value, which is the weighted average of precision and recall, our model shows 95.00% accuracy in binary classification having the best outcome compared to the existing approaches.

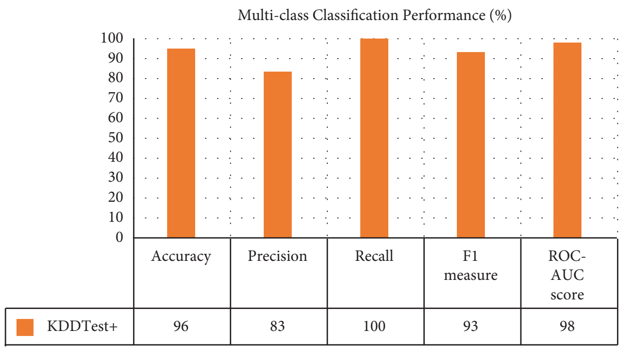
4.3.2. For Multiclass Classification. Our proposed model shows excellent results by offering a recall of 100%, and an accuracy of 96.00% in multiclass classification on the NSL-KDDTest + datasets and a ROC-AUC score of 98%. Only 83.85% accuracy was obtained for the NSL-KDDTest + dataset by Hou et al. [8] who applied an HLSTM-IDS. In Yin et al. [10], the authors used an RNN-IDS and claimed an accuracy of 81.29% for multiclass classification on the NSL-KDD dataset. They did not mention the value of recall and precision. Ait Tchakoucht and Ezziyyani [11] claimed that their RNN-MLESM approach achieved an accuracy of 81% when performing the 4-class classification on the NSL-KDDTest + dataset. Shone et al. [12], the author used an S-NDAE and achieved an accuracy of 85.42%. Also, Le et al. [13] claimed their LSTM model achieved an accuracy of 92.00%. Imrana et al. [14] used BiDLSTM methods, achieving recall, F1-score, and accuracy, respectively, of 91.36%, 91.67%, and 91.36%. Kunang et al. [15] developed DAE + DNN with an accuracy of 83.83%. Imrana et al. [17] developed Chi-square BidLSTM, achieving an accuracy of 95.62%. This comparative discussion shows that our proposed model showed superior results compared to all methods cited in Table 2.

4.3.3. Limitation. The outcome of the experiments shows that the proposed method can be applied to successfully identify intrusion in network systems. However, compared to the existing approaches, the proposed method's multi-layer LSTM model is more sophisticated due to class



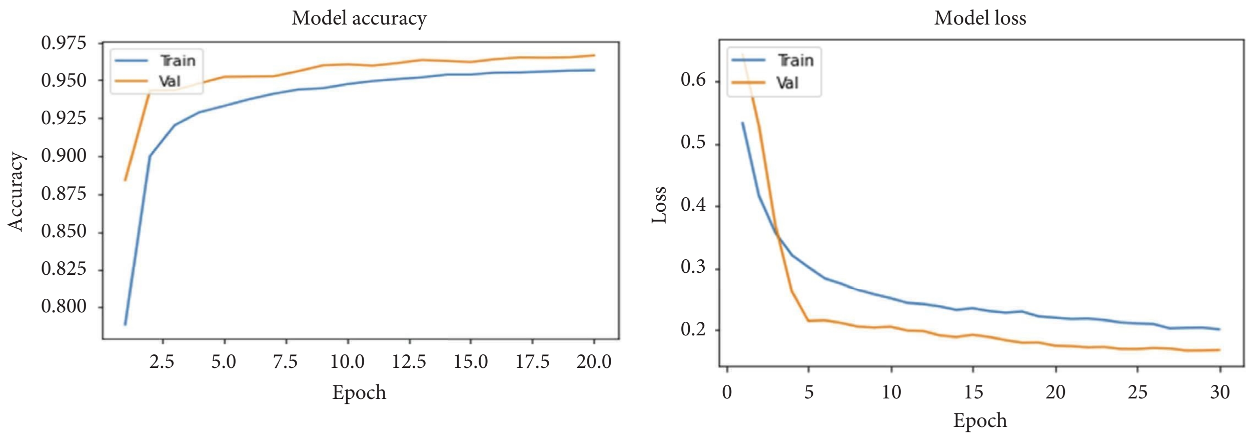
(a) (b)

Figure 6: (a) Training accuracy and validation accuracy and (b) training loss and validation loss of the proposed model for binary classification.



	Accuracy	Precision	Recall	F1 measure	ROC-AUC score
🟧 KDDTest+	96	83	100	93	98

Figure 7: Performance of the proposed model for multiclass classification.



(a) (b)

Figure 8: (a) Training accuracy and validation accuracy and (b) training loss and validation loss of the proposed model for multiclass classification.

Table 1: Comparison with the state of arts for binary classification.

Reference	Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Yin et al. [10]	RNN	83.28	—	—	—
Ait Tchakoucht and Ezziyyani [11]	RNN-MLESM	83.00	—	—	—
Imrana et al. [14]	BiDLSTM	94.26	**99.05**	90.79	94.74
Rajesh Kanna and Santhi [16]	OCNN-HMLSTM	90.67	86.71	**95.19**	91.46
This work	Multilayer LSTM	**95.00**	95.00	95.00	**95.00**

Table 2: Comparison with the state of arts for multiclass classification.

References	Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Hou et al. [8]	HLSTM-IDS	83.85	—	—	—
Yin et al. [10]	RNN	81.29	—	—	—
Ait Tchakoucht and Ezziyyani [11]	RNN-MLESM	81.00	—	—	—
Shone et al. [12]	S-NDAE	85.42	**100.00**	85.42	87.37
Le et al. [13]	LSTM	92.00	—	—	—
Imrana et al. [14]	BiDLSTM	91.36	92.81	91.36	91.67
Kunang et al. [15]	DAE + DNN	83.83	86.02	83.83	82.04
Imrana et al. [17]	Chi-square BidLSTM	95.62	—	—	**95.65**
This work	Multilayer LSTM	**96.00**	83.00	**100.00**	93.00

imbalanced issues that require more training time. Moreover, we have obtained the lowest precision score of 83% in multiclass classification on the NSL-KDDTest + datasets. These facts are the key restrictions of this work.

5. Conclusion

In this paper, an optimum multilayer architecture is built in order to attain maximum accuracy by employing stacking multiple layers of LSTM cells in a more effective approach. It possesses better stability and performs consistently. The other aspects of the proposed technique include the effective enhancement of feature selection, the prevention of model overfitting by dropout layers and batch normalization, the capability to eliminate lower correlation, the ability to handle categorical features by one-hot encoding, and the capacity to reduce error by using the backpropagation technique. The performance of the proposed model presents an accuracy of 95% for binary classification and an accuracy of 96% for multiclass classification on the NSL-KDDTest + datasets. The proposed technique is able to detect the majority of the attacks with a 98% ROC-AUC score both in binary and multiclass classification on the NSL-KDDTest + datasets. Our optimum multilayer architecture would be implemented on real-time applications due to dealing with authentic datasets and achieving great performance. For future works, we may prepare a customized dataset and elaborate a hybrid model, addressing the class imbalanced issues for intrusion detection for real-world implementation and simulation. In conclusion, deep learning analysis in the field of cyber security is still complex, however, the overall performance evaluation metrics of our proposed model are robust and effective compared to other relevant approaches.

Data Availability

The data used to support the findings of this study are included in the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] T. S. G. Halbouni and M. H. Habaebi, M. H. M. Kartiwi and R. Ahmad, CNN-LSTM: hybrid deep neural network for network intrusion detection system," *IEEE Access*, vol. 10, 2022.

[2] D. N. Mhawi, A. Aldallal, and S. Hassan, "Advanced feature-selection-based hybrid ensemble learning algorithms for network intrusion detection systems," *Symmetry*, vol. 14, no. 7, p. 1461, 2022.

[3] S. Y. Diaba, M. Shafie-khah, and M. Elmusrati, "On the performance metrics for cyber-physical attack detection in smart grid," *Soft Computing*, vol. 26, no. 23, pp. 13109–13118, 2022.

[4] M. Chalé and N. D. Bastian, "Generating realistic cyber data for training and evaluating machine learning classifiers for network intrusion detection systems," *Expert Systems with Applications*, vol. 207, Article ID 117936, 2022.

[5] I. Ullah and Q. H. Mahmoud, "Design and development of RNN anomaly detection model for IoT networks," *IEEE Access*, vol. 10, pp. 62722–62750, 2022.

[6] F. Laghrissi, S. Douzi, K. Douzi, and B. Hssina, "Intrusion detection systems using long short-term memory (LSTM)," *Journal of Big Data*, vol. 8, no. 1, pp. 65–16, 2021.

[7] Y. Fu, Y. Du, Z. Cao, Q. Li, and W. Xiang, "A deep learning model for network intrusion detection with imbalanced data," *Electronics*, vol. 11, no. 6, p. 898, 2022.

[8] H. Hou, Y. Xu, M. Chen et al., "Hierarchical long short-term memory network for cyberattack detection," *IEEE Access*, vol. 8, pp. 90907–90913, 2020.

[9] S. Naseer, Y. Saleem, S. Khalid et al., "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[10] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[11] T. Ait Tchakoucht and M. Ezziyyani, "Multilayered Echo-State Machine: a Novel architecture for efficient intrusion detection," *IEEE Access*, vol. 6, pp. 72458–72468, 2018.

[12] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[13] T. T. H. Le, Y. Kim, and H. Kim, "Network intrusion detection based on novel feature selection model and various recurrent neural networks," *Applied Sciences*, vol. 9, no. 7, p. 1392, 2019.

[14] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf, "A bidirectional LSTM deep learning approach for intrusion detection," *Expert Systems with Applications*, vol. 185, Article ID 115524, 2021.

[15] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization," *Journal of Information Security and Applications*, vol. 58, Article ID 102804, 2021.

[16] P. Rajesh Kanna and P. Santhi, "Unified deep learning approach for efficient intrusion detection system using integrated spatial–temporal features," *Knowledge-Based Systems*, vol. 226, Article ID 107132, 2021.

[17] Y. Imrana, Y. Xiang, L. Ali et al., "χ 2-bidlstm: a feature driven intrusion detection system based on χ 2 statistical model and bidirectional lstm," *Sensors*, vol. 22, no. 5, p. 2018, 2022.

[18] A. Li and S. Yi, "Intelligent intrusion detection method of industrial internet of things based on cnn-biLSTM," *Security and Communication Networks*, vol. 2022, Article ID 5448647, 8 pages, 2022.

[19] S. Shende and S. Thorat, "Long short-term memory (LSTM) deep learning method for intrusion detection in network security," *International Journal of Engineering Research*, vol. 9, 2020.