Scientific Research Publishing

International Journal of Internet and Distributed Systems, 2024, 6, 25-39
https://www.scirp.org/journal/ijids
ISSN Online: 2327-7165
ISSN Print: 2327-7157

Deep Learning-Based Two-Step Approach for Intrusion Detection in Networks

Kamagaté Beman Hamidja[1], Kanga Koffi[1], Kouassi Adless[1], Olivier Asseu[1,2], Souleymane Oumtanaga[3]

[1]Ecole Supérieure Africaine des Technologies de l'Information et de la Communication (ESATIC), LASTIC Laboratory of ESATIC, Abidjan, Côte d'Ivoire
[2]UMRI STI, INPHB INPUMRI STI, INPHB INP-Houphouët Boigny, Yamoussoukro, Côte d'Ivoire
[3]UMRI MSN INPHB INP-Houphouët Boigny, Yamoussoukro, Côte d'Ivoire
Email: beman.kamagate@esatic.edu.ci

How to cite this paper: Hamidja, K.B., Koffi, K., Adless, K., Asseu, O. and Oumtanaga, S. (2024) Deep Learning-Based Two-Step Approach for Intrusion Detection in Networks. *International Journal of Internet and Distributed Systems*, **6**, 25-39.
https://doi.org/10.4236/ijids.2024.62002

Received: May 2, 2024
Accepted: May 25, 2024
Published: May 28, 2024

Abstract

Intrusion Detection Systems (IDS) are essential for computer security, with various techniques developed over time. However, many of these methods suffer from high false positive rates. To address this, we propose an approach utilizing Recurrent Neural Networks (RNN). Our method starts by reducing the dataset's dimensionality using a Deep Auto-Encoder (DAE), followed by intrusion detection through a Bidirectional Long Short-Term Memory (BiLSTM) network. The proposed DAE-BiLSTM model outperforms Random Forest, AdaBoost, and standard BiLSTM models, achieving an accuracy of 0.97, a recall of 0.95, and an AUC of 0.93. Although BiLSTM is slightly less effective than DAE-BiLSTM, both RNN-based models outperform AdaBoost and Random Forest. ROC curves show that DAE-BiLSTM is the most effective, demonstrating strong detection capabilities with a low false positive rate. While AdaBoost performs well, it is less effective than RNN models but still surpasses Random Forest.

Keywords

Cybersecurity, CICIDDS2017, Intrusion Detection, BiLSTM, Deep Auto-Encoder

1. Introduction

The security of computer systems is a sensitive and worrying issue. The spectacular progress of information and communication technologies today offers inescapable facilities for file transfer, messaging and many other forms of information exchange. The development of computerized exchanges has unfortunately been

accompanied by the development of malicious activities whose motives are as numerous as they are dangerous, and which evolve over time [1]. Taking advantage of the growing connectivity of information technology systems, particularly to the Internet, the possibilities for remote attacks are greater and more threatening.

Intrusion Detection Systems (IDS) can be set up to detect any attempt to violate security mechanisms, by monitoring systems on an ongoing basis. Intrusion detection involves scanning network traffic, collecting all events, analyzing them and generating alarms if malicious attempts are identified [2]. Intrusion Detection Systems (IDS) are essential for cybersecurity. They detect threats before they cause damage, block intrusions by monitoring network activities in real-time, and protect sensitive data from unauthorized access, especially in critical sectors. By identifying vulnerabilities, they help strengthen system resilience and ensure compliance with security standards, facilitating audits. These functions make IDS essential tools for proactive cybersecurity. They are widely deployed across IT systems and have become integral to the design of security strategies. They are generally used to monitor access and information flow, with the aim of determining any malicious behavior, whether from inside or outside the information system, and making this information available to security administrators. As an option, intrusion detection systems can react to malicious behavior and take countermeasures.

In recent years, a range of approaches has been developed to enhance intrusion detection systems, with a focus on both rule-based methods and machine learning techniques. Rule-based systems are relatively straightforward to implement and are effective at identifying known attacks while maintaining a low rate of false positives [3]. However, they have limitations when it comes to detecting unknown attacks. On the other hand, machine learning methods are capable of identifying novel threats, such as "zero-day" attacks, but they come with their own set of challenges. Although machine learning-based IDS can be effective in recognizing new threats [4], they may also produce a high number of false positives and false negatives. This means they might incorrectly label legitimate traffic as malicious or fail to detect genuinely suspicious traffic, potentially undermining the overall reliability of network protection mechanisms.

In this work, we propose a two-tier approach to tackle these challenges. The first tier utilizes deep autoencoders to compress the training dataset, focusing on extracting only the most relevant features. The second tier utilizes a Bidirectional Long Short-Term Memory (BiLSTM) network, a type of Recurrent Neural Network (RNN), to process the compressed dataset from the first tier and classify traffic as either normal or anomalous. The BiLSTM stands out for its bidirectional processing, capturing both past and future contexts to enrich the understanding of event relationships, essential for detecting complex intrusion patterns. Its long-term memory enables it to retain information over extended sequences, useful for spotting abnormal behaviors in network traffic. Finally, its adaptability to time series makes it an effective real-time model, responsive to emerging threats. These features enhance its accuracy and effectiveness in proactive intrusion detection [5].

The subsequent sections of this work are organized as follows: Section 2 provides a review of related work, Section 3 outlines the methodology and proposed framework, and Section 4 covers the experimentation, results, and discussion. The paper concludes with a summary and recommendations for future research.

2. Related Work

For intrusion detection, the first approach used in the literature is the rule-based approach. It compares the signature of incoming flows to a range of signatures stored in a database that are already identified as malicious. If a signature matches one in the database, it indicates that the traffic may be malicious. This approach is widely developed in many research papers.

In paper [6], the authors introduce a signature-based intrusion detection system (IDS) designed to detect denial-of-service (DoS) and routing attacks in IoT networks. The major innovation of this IDS is its hybrid configuration, which combines centralized and distributed components. The detection module is installed on the main router, while lightweight modules are positioned near IoT devices to monitor and report traffic. The work [7] analyzes intrusion detection systems based on fuzzy logic, which are designed using various data mining techniques. Thanks to fuzzy logic, these systems can better account for the uncertainty and ambiguity inherent in intrusion detection data, thus providing more effective management of complex situations where attack patterns are not always clear or well-defined. In [8], the authors performed an experimental study to evaluate the detection capabilities of signature-based intrusion detection systems in the context of web attacks. The results show that the predefined configuration rulesets of generic solutions like Snort, ModSecurity, and Nemesida Free offer lower-than-expected detection performance for known attacks, even when set to the most sensitive configurations. From these works, it can be concluded that although rule-based detection systems are effective in identifying known intrusions, their performance can be significantly impacted by the system configurations. Furthermore, these systems are unable to detect threats whose signatures are not already listed in their database.

The second one, behavioral approach assumes that normal activity is different from intrusive activity. All that's needed is a profile for normal activity, and a mechanism for comparing current activity with the established profile to detect significant deviations that will be considered as possible intrusions.

Study [9] presents a behavioral approach to intrusion detection that combines Accelerated Particle Swarm Optimization (APSO) with a Support Vector Machine (SVM) to create an IDS model. The simulation results demonstrate a significant improvement in performance. Compared to other methods applied to the same dataset, the proposed model achieves superior detection accuracy.

To address the challenges posed by the dynamic and constantly evolving nature of malicious attacks, which also occur in large volumes and require scalable solutions, study [10] focuses on using Deep Neural Networks to detect malware and

malicious behavior at both the network and host levels. Experiments conducted on various datasets demonstrate a high level of accuracy in intrusion detection. Study in [11].

In work [12], the authors emphasize the increasing difficulty in accurately detecting intrusions due to the evolving nature of cyber-attacks, which pose significant risks to data confidentiality, integrity, and availability. After evaluating several machine learning algorithms, which, despite being capable of detecting unknown attacks, produced many false positives, the study concludes that decision trees are the most effective among machine learning algorithm, delivering superior performance in intrusion detection. As a result, an optimized framework was developed based on this algorithm.

Compared to the previous signature-based approach, this method can detect unknown attacks, but it tends to produce a higher number of false positive alarms and requires more time to reach convergence, especially when there are many features. To address these shortcomings, we suggest employing a two-step deep learning approach for intrusion detection. Initially, we utilize a stacked autoencoder to reduce the dimensionality of features. Subsequently, the resultant vector serves as input to a Bidirectional Long Short-Term Memory to detect whether the behavior is normal or abnormal.

3. Materials and Methods

This section outlines our methodological approach to intrusion detection. It comprises two phases: dimensionality reduction and classification. Dimensionality reduction step help as to select less features than features in the original database. Then at second step we apply classification to vector of the less selected features.

3.1. Dimensionality Reduction with Deep Auto-Encoder

Instance classification encounters several problems, such as large number of features. In order to improve classification accuracy and reduce computation times, we need to reduce the dimensionality of the data. For that end, we use Deep Auto-Encoder (DAE) [13]. Contrary to the conventional Ordinary Auto-Encoder (OAE) with a single hidden layer, the Deep Auto-Encoder (DAE) depicted in **Figure 1** utilizes multiple hidden layers.

The hidden layers link the input layer with the output layer. They are stacked one after the other to form the encoder with the input layer, and the decoder with the output layer. In order to extract the strongest and most important features for proper classification, the DAE goes through two main phases: encoder and decoder. The encoding phase takes place between the input layer and the hidden layers up to the bottleneck layer. In this phase, the input data is transformed into reduced-dimensional representation, while retaining relevant features.

To train the autoencoder, we began by determining the number of hidden layers and neurons per layer, monitoring the risk of overfitting. Next, we selected the activation function, opting for one that offers both speed and stability. Regarding

the cost function, in line with the literature [13] recommending mean squared error (MSE) for reconstructions, we adopted this approach. For the learning rate, we started with a low value, which we gradually adjusted to avoid overly slow convergence. We then determined the appropriate number of epochs and batch size to stabilize the model. The final step was to assess the quality of the reconstruction on a validation set, which led us to adjust parameters such as the number of layers, the learning rate, and the batch size.

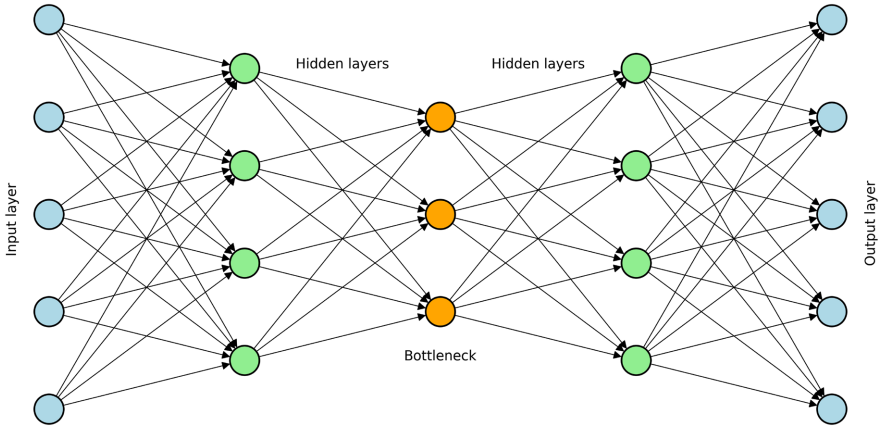

Figure 1. Deep Auto-Encoder (DAE) with hidden layers.

The encoder evaluates (formula 1) vector h (latent or bottleneck vector) with size m from the input vector X which represents the original data.

$$h = f\left(W^{(1)}X + b^{(1)}\right) \tag{1}$$

where: $W^{(1)}$ is a matrix $m \times n$, $b^{(1)}$ is a bias vector of size n and $f(\,)$ is an encoder activation function.

After that, came the decoder phase. It takes place between the hidden layers (starting with the bottleneck layer) and the output layer which has the same number of nodes as the input layer. The hidden layers in this phase are symmetrical to the hidden layers in the previous phase, with one in common (the bottleneck layer). The DAE try to have an output vector $\hat{X}$ that is closet possible representation of the initial attributes (those of the input layer: X), by minimizing the loss function. In this work loss function is mean squared error loss function (formula (2)).

$$\text{Loss} = \frac{1}{n}\left(\sum_{i=1}^{n}\left(X_i - \hat{X}_i\right)^2\right) \tag{2}$$

The decoder tries to build output vector from the latent vector h, the result of which is the vector $\hat{X}$.

$$\hat{X} = g\left(W^{(2)}h + b^{(2)}\right) \tag{3}$$

where $W^{(2)}$ is a matrix $n \times m$, $b^{(2)}$ is a bias vector of size n, $\hat{X}$ is a vector of size m, a reconstruction of the input vector X as close as possible to it.

3.2. Data Classification with BiLSTM

After remarkably reduced the number of attributes (features) in the original database, in this section, we propose to predict the classes of the new instances, for which we've chosen to apply Bidirectional Long Short-Term Memory (BiLSTM) [14]. BiLSTM architecture is an extension of recurrent neural networks (RNN), designed to capture temporal dependencies in data sequences in both past and future directions. It is composed with two Long Short-Term Memory (LSTM) [15].

A sequence of LSTM [16] is made up of cells connected across time steps *t*. Each cell's output is regulated by a group of activation functions $\{f(t), i(t), o(t)\}$, referred to as gates [17], as depicted in **Figure 2**. These gates yield values in $\mathbb{R}^T$, where T represents the dimension (the number of cells in the sequence) of the LSTM. The function $f(t)$, often referred to as the forget gate, determines the degree to which the information from the preceding cell c_{t-1} is discarded [18]. The input gate $i(t)$ specifies the portion of information to be stored in cell c_t. Meanwhile, the output gate $o(t)$ governs the fraction of the internal state transmitted to the subsequent cell.

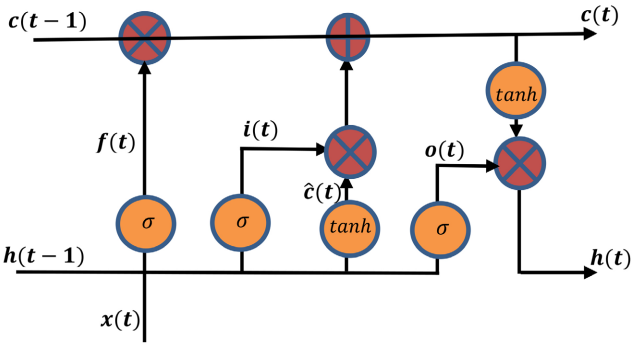

Figure 2. Composition of the LSTM cell.

The cell input $x(t)$ at time t concatenates with output of a cell $c(t-1)$ at previous time step $(t-1)$ which is $h(t-1)$. The resultant vector traverses through the input node and passes through the gates responsible for input, forgetting, and output to give $h(t)$. Evaluation of $h(t)$ (Equation (4.6)) for each cell can be find in many papers, for example in [19] and [20].

$$f(t) = \sigma\left(w_{fh}h(t-1) + w_{fx}x(t) + b_f\right) \tag{4.1}$$

$$i(t) = \sigma\left(w_{ih}h(t-1) + w_{ix}x(t) + b_i\right) \tag{4.2}$$

$$\tilde{c}(t) = \tanh\left(w_{\tilde{c}h}h(t-1) + w_{\tilde{c}x}x(t) + b_{\tilde{c}}\right) \tag{4.3}$$

$$c(t) = f(t)\cdot c(t-1) + i(t)\cdot \tilde{c}(t) \tag{4.4}$$

$$o(t) = \sigma\left(w_{oh}h(t-1) + w_{ox}x(t) + b_o\right) \tag{4.5}$$

$$h(t) = o(t)\cdot \tanh\left(c(t)\right) \tag{4.6}$$

It is important to notice that, there is many variants of LSTM cell depend on how function σ and tanh are arranged and their number in the cell [18]. An LSTM layer operates unidirectionally, processing input sequentially from left to right, thus encoding dependencies solely on preceding elements in the sequence. To address this limitation, we employ an additional LSTM layer that operates in the opposite direction, enabling the detection of dependencies on subsequent elements in the text by processing input from right to left. This arrangement gives rise to a neural network known as a Bidirectional LSTM (BiLSTM) [21]. It enables the network to assign equal significance to both the initial and final segments of the sequence, leading to enhanced performance.

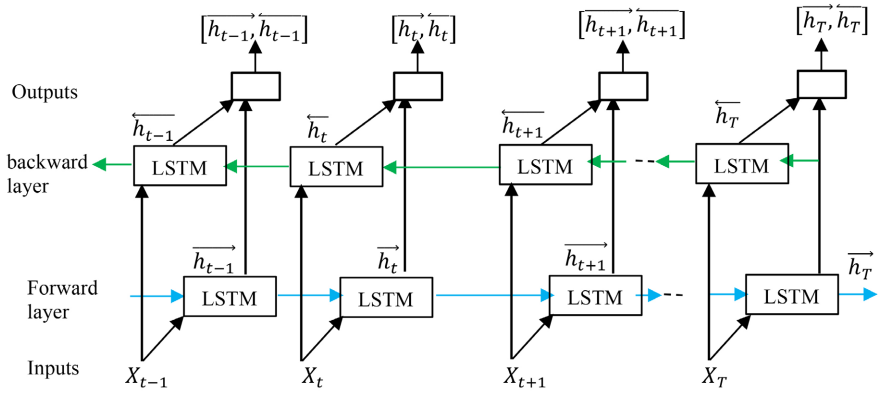

Figure 3. Architecture of a bidirectional LSTM (BiLSTM) layer.

Additionally, we illustrate an instance of a bidirectional LSTM layer in **Figure 3**. The result of a BiLSTM layer is the concatenation of the outputs in both directions, as show in Equation (5)

$$H = \left[\overrightarrow{h_T}, \overleftarrow{h_T}\right] \tag{5}$$

where $\overrightarrow{h_T}$ and $\overleftarrow{h_T}$ are define by Equation (4.6).

3.3. Processing Framework

The intrusion detection method we propose is divided into two steps. The first step involves dimensionality reduction using Deep Auto-Encoder (DAE), followed by a phase of behavior class prediction with BiLSTM recurrent neural networks. The dimensionality reduction process begins with data cleaning, which involves identifying, correcting, or removing erroneous data. Errors are indicated by missing values (NaN), infinite values (Inf), and empty cells within the dataset. To address these issues, we have developed an algorithm that replaces NaN values with 0, substitutes Inf values with the maximum value of the dataset, and fills empty cells with the mean value of the relevant attributes.

Following data cleaning, we moved on to normalization. This preprocessing step adjusts the data values to ensure they fall within a consistent range, which helps machine learning algorithms function more efficiently and reliably. In this study, we applied Z-Score normalization) as our chosen technique. This method

standardizes the data to have a mean of 0 and a standard deviation of 1, as specified in Equation (6).

$$X' = (X - \mu)/\sigma \tag{6}$$

where X represents the original data value, X' is the normalized value, with μ and σ representing the mean and standard deviation, respectively. Z-Score normalization is the most appropriate method for autoencoders [22] as it provides a uniform and centered scale for the data, which enhances the efficiency of the autoencoder's reconstruction and the model's stability. The final step in this phase was to design and train the deep autoencoder architecture to extract reduced and representative features from the original data. **Algorithm 1** below outlines the sequence of steps involved in the dimensionality reduction process.

Algorithm 1. Dimensionality reduction.

1	Input: D (original Dataset)
2	Length = row number of D
3	Output: Compressed Dataset
4	For ($i \leftarrow 1$ to $i \leq$ Length)
5	If $D_i(x_1, x_2, \cdots, x_n)$ contain inf, Nan or empty replace
6	respectively with max value, zero and maximum value
7	End if
8	End for
9	Update D
10	Apply Data normalization and update D (Equation (6))
11	Build _the _DAE (input data :D, Number of layers, activation function):
12	For each layer in the encoder phase: Apply $h(l) = f(W(l) * h(l-1) + b(l))$
13	For each layer l in the decoder phase: Apply $h(l) = f(W(l) * h(l-1) + b(l))$
14	Train_the_DAE (input vector X in D, encoder network, decoder network, Epoch, Batch size):
15	Z = encoder(X) (Equation (1))
16	X' = decoder(Z) (Equation (3))
17	Loss = MeanSquaredError (X, X') (Equation (2))
18	Compressed dataset:
19	Update D with latent vector Z

In line 12 and 13, $h(0)$ is the input data. $f(.)$ is activation function. $W(l)$ and $b(l)$ are the weights and biases of layer l.

The compressed data from Algorithm 1 is subsequently utilized in the classification phase, which is based on a Bidirectional LSTM, to determine if the traffic is normal or indicative of an attack. The steps of this process are outlined in Algorithm 2.

Algorithm 2. Intrusion detection.

Input: $D=\left[X_1, X_2, \cdots, X_n\right]$ (Compressed data from Algorithm 1)

Output: Class of traffic (normal or abnormal)

Split Data into training and testing set:

$X_{train} = X_1,\ X_2, \cdots, X_k$

$X_{test} = X_{k+1}, X_{k+2}, \cdots, X_n$

Bulding the model:

  Forward LSTM:

    For each time step t from 1 to T

      For each X_i in X_{train} do:

        Calculate $f(t)$ (Equation (4.1)), $i(t)$ (Equation (4.2) and $\tilde{c}(t)$ (Equation (4.3))

        Update Cell state $c(t)$ (Equation (4.4))

        Calculate $o(t)$ (Equation (4.4)) and $\overrightarrow{h(t)}$ Equation (4.5))

  Backward LSTM:

    For each time step t from T to 1

      For each X_i in X_{train} do

        Calculate $f(t)$ (Equation (4.1)), $i(t)$ (Equation (4.2) and $\tilde{c}(t)$ (Equation (4.3))

        Update Cell state $c(t)$ (Equation (4.4))

        Calculate $o(t)$ (Equation (4.4)) and $\overleftarrow{h(t)}$ Equation (4.5))

  Concatenation:

    For each time step t, concatenate the forward and backward LSTM hidden states

      $h(t)=\left[\overrightarrow{h(t)}, \overleftarrow{h(t)}\right]$ (Equation (5))

Output Layer: $\hat{y}_t = \sigma\left(Wh(t)\right)$ (σ is an activation function, W is the weight matrix)

Training:

  Use the loss function on $\hat{y}_t$ to compute the error

  Optimize the weight of $\hat{y}_t$ to reduce loss

Evaluation:

  Test the model on the validation data X_{test}

4. Experimentation, Results and Discussion

4.1. Experimentation Framework AND Results

The experiments were carried out on a PC equipped with a 16 GB RAM Core i7 processor running Windows 11 64-bit. For software, we utilized Python 3.7 within the Anaconda integrated development environment (IDE), which serves as the editor and includes various libraries such as Scikit-learn, TensorFlow, Keras, SciPy, NumPy, Pandas, and Matplotlib.

Concerning the dataset, we use a most recent Canadian Institute for Cybersecurity (CIC) dataset called CIC2017 available at [23]. The CICIDS2017 dataset offers a wide range of attack scenarios, including Denial of Service (DoS), Distributed Denial of Service (DDoS), brute force, web attacks, infiltration, botnet, and others. This variety makes it ideal for testing and comparing different intrusion detection techniques. The dataset simulates realistic network traffic by combining both normal and malicious activities, closely resembling real-world conditions as opposed to fully synthetic datasets. It provides over 80 features extracted from network traffic, encompassing basic information like IP addresses and port numbers, as well as more advanced statistical metrics such as packet length and flow duration. Further details can be found at [24].

As mentioned in our proposal, we have chosen deep auto-encoder as module for dimensionality reduction. After carrying out several executions, to find the right optimal value for the number hidden layers, we return 3 hidden layers between the input layer which is a vector of size 80 and the bottleneck layer whose size is 25. The total number of trainable parameters is 22,141. The learning curve for our deep autoencoder was set to 20 epochs with a batch size of 320 (after several evaluation tests). Table 1 summarizes the features of DAE.

Table 1. Characteristics deep auto-encoder.

Steps		Layer type	Output size (features)	Number of parameters
Encoding		Input	80	0
		Hidden_1	64	5184
		Hidden_2	50	3250
		Hidden_3	34	1734
		Bottleneck	25	875
	Decoding	Hidden_4	34	884
		Hidden_5	50	1750
		Hidden_6	64	3264
		output	80	5200

The proposed DAE enabled us to reduce the number of attributes from 80 to 25 (*i.e.* a compression factor of 1:3.2). Figure 4 shows the graph of the loss function for the different number of epochs, and this in both train and test dataset. As the amount of loss is under 5×10^{-4} for epoch near 40. An output vector very close to the input vector is obtained from the 25 selected features. Then, BiLSTM help us to classify data instances with satisfactory accuracy.

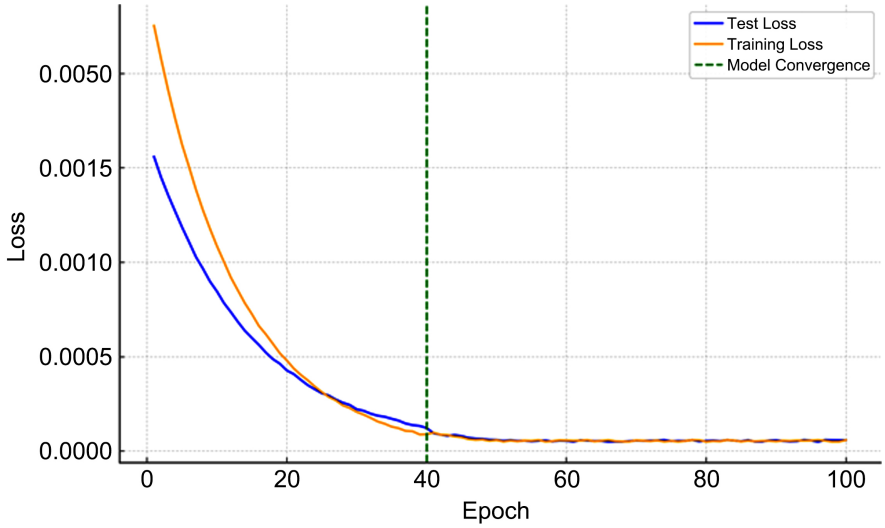

Figure 4. Training and test loss over 100 epochs.

We also conducted another experiment to evaluate the performance of our proposal. For this, we used two metrics: accuracy and recall. Our proposal (DAE-BiLSTM) performs better than random forests, AdaBoost, or BiLSTM (see Figure 5). DAE-BiLSTM achieved an accuracy of 97% and a recall of 95%, respectively.

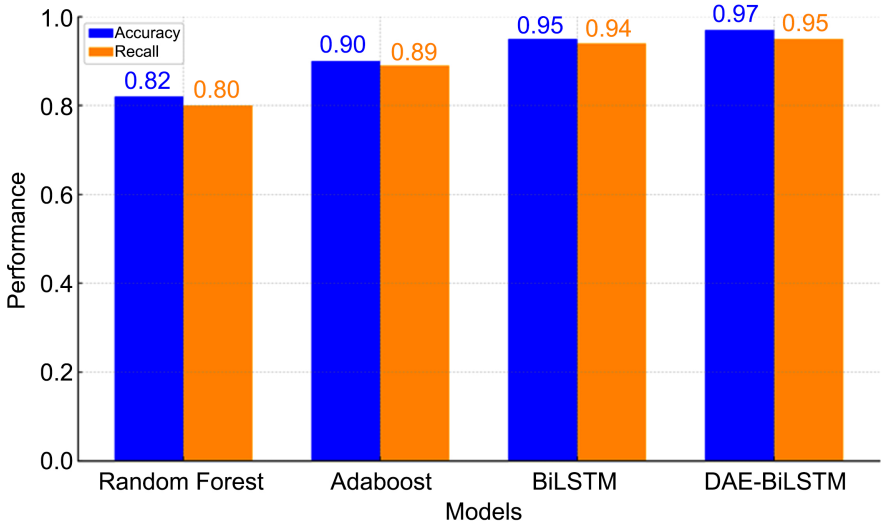

Figure 5. Performance metrics (accuracy and recall) of Randon Forest, Adaboost, BiLSTM DAE-BiLSTM.

Finally, we evaluated the ROC curves and AUC values of Random Forest, Adaboost, BiLSTM, and DAE-BiLSTM in an intrusion detection task. The results, shown in Figure 5, indicate that DAE-BiLSTM achieved the highest AUC, close to 0.93, outperforming all other models.

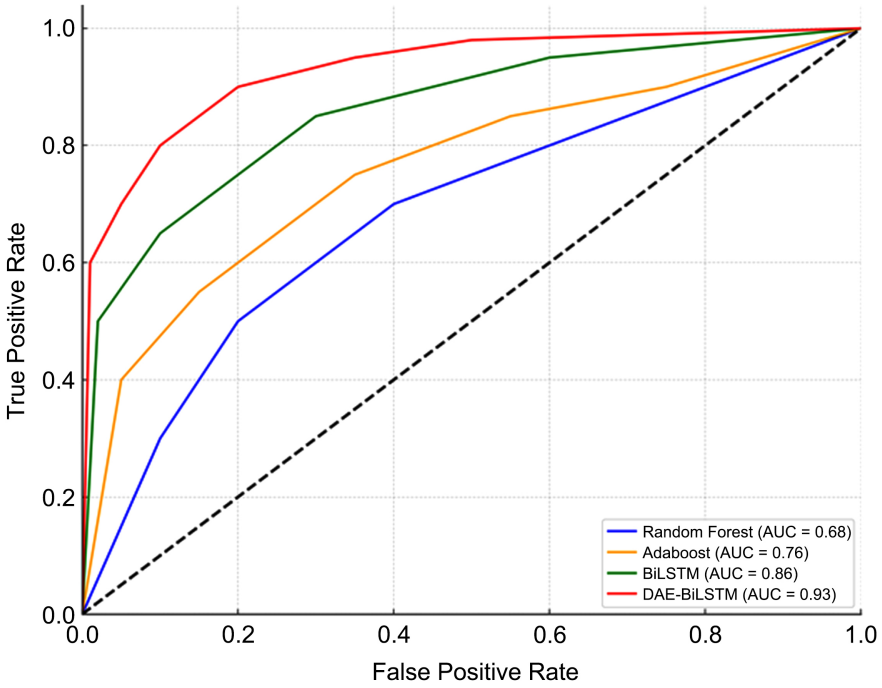

Figure 6. ROC Curves for various intrusion detection model.

4.2. Discussion

Upon examining Figure 4, we observe that the training and test loss curves gradually decrease before stabilizing around the 40th epoch, reaching a loss rate significantly below 5×10^{-4}. At this point, the two curves converge, indicating that the model is learning effectively from the training data while generalizing properly to the test data. The progressive decrease in loss followed by stabilization at a low value reflects the model's good performance. This convergence toward a low loss value, with no sign of divergence between the two curves, suggests that there is neither underfitting nor overfitting. The model successfully captures the characteristics of the data without overfitting the training set, which is an indicator of its generalization capability.

With regard to the histogram in Figure 5, which represents the models' performance on Accuracy and Recall, we note that the DAE-BiLSTM model is the most performant for this intrusion detection problem, with higher performance on both metrics. The BiLSTM follows closely, while Adaboost and Random Forest are less effective. This indicates that, for this complex problem, recurrent neural networks are a top choice. More specifically, the fact that DAE-BiLSTM has the highest Accuracy (0.97) and Recall (0.95) among the four models means that it is not only capable of making much more accurate predictions than the others, but

it is also excellent at correctly identifying positive cases. Therefore, it is a model that generalizes well.

The performance of the DAE-BiLSTM compared to other models can be explained by the fact that deep autoencoders improve data processing by reducing dimensionality, learning compact representations that highlight key features and filter out noise. This enhances the quality of input features for the BiLSTM and boosts detection accuracy. By focusing on essential characteristics, the DAE-BiLSTM reduces the risk of overfitting and improves generalization, enabling better identification of network attacks in new data. Lower input dimensionality also reduces computational costs, allowing faster training and more efficient resource use, while smoothing the optimization process and decreasing the likelihood of the model getting stuck in local minima, which potentially boosts accuracy. The refined features from the first step help the BiLSTM in the second step better distinguish between normal and malicious activities, reducing false positives and improving the overall reliability of the intrusion detection system.

Figure 6 also shows the DAE-BiLSTM's superiority, as its Receiver Operating Characteristic (ROC) curve approaches the top-left corner, demonstrating outstanding intrusion detection accuracy with minimal errors. This model excels by combining the strengths of the Deep Autoencoder (DAE) for dimensionality reduction and the BiLSTM for capturing temporal patterns, resulting in exceptional performance. Even when used on its own, BiLSTM performs well, indicating that recurrent neural networks are effective for intrusion detection due to their ability to recognize temporal relationships in data. On the other hand, traditional models like Adaboost and Random Forest fall short, likely because they struggle to handle complex, high-dimensional data. Area Under the Curve (AUC) serves as a crucial metric for evaluating model performance, with higher values indicating better class separation. The DAE-BiLSTM leads the pack with an AUC of 0.93, followed by BiLSTM (0.86), Adaboost (0.76), and Random Forest (0.68). The success of DAE-BiLSTM lies in its dual approach: dimensionality reduction via the DAE, which filters noise and emphasizes important features, and temporal pattern recognition by the BiLSTM. This combination enhances the model's generalization and reliability, making DAE-BiLSTM the optimal choice for intrusion detection tasks.

5. Conclusions

In this study, we present a two-step approach for intrusion detection in computer systems, leveraging the advantages of deep autoencoders for dimensionality reduction and BiLSTM for capturing detailed temporal patterns in the data. In the first step, the deep autoencoder reduces the dimensionality of the input features, filtering out noise and retaining only the most relevant characteristics. In the second step, the refined feature vector serves as input to the BiLSTM, which further learns the temporal dependencies and patterns, enhancing the model's ability to detect intrusions effectively.

This combined approach has yielded promising results, outperforming several

existing methods in the literature. The dimensionality reduction not only improves the quality of the input data but also accelerates the learning process, potentially leading to faster convergence during training. Future work will focus on validating this observation and exploring alternative deep learning architectures for the second stage to identify the most suitable model for generalization. By integrating dimensionality reduction with advanced temporal pattern recognition, the proposed method offers a robust and efficient solution for intrusion detection, demonstrating significant improvements in accuracy and reliability.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Rai, A., *et al.* (2020) A Review of Information Security: Issues and Techniques. *International Journal for Research in Applied Science and Engineering Technology*, **8**, 953-960. https://doi.org/10.22214/ijraset.2020.5150

[2] Alanazi, H., Noor, R., Zaidan, B.B., *et al.* (2010) Intrusion Detection System: Overview. *Journal of Computing*, **2**, 130-133. https://doi.org/10.48550/arXiv.1002.4047

[3] Salih, R., Den Hartog, J. and Smulders, E. (2020) Semantical Rule-Based False Positive Detection for IDS. https://pure.tue.nl/ws/portalfiles/portal/174214825/Salih_R..pdf

[4] Santhosh Kumar, S.V.N., Selvi, M. and Kannan, A. (2023) A Comprehensive Survey on Machine Learning-Based Intrusion Detection Systems for Secure Communication in Internet of Things. *Computational Intelligence and Neuroscience*, **2023**, Article ID: 8981988. https://doi.org/10.1155/2023/8981988

[5] Siami-Namini, S., Tavakoli, N. and Namin, A.S. (2019). The Performance of LSTM and BiLSTM in Forecasting Time Series. 2019 *IEEE International Conference on Big Data* (*Big Data*), Los Angeles, 9-12 December 2019, 3285-3292. https://doi.org/10.1109/bigdata47090.2019.9005997

[6] Ioulianou, P., Vasilakis, V., Moscholios, I., *et al.* (2018) A Signature-Based Intrusion Detection System for the Internet of Things. *Information and Communication Technology Form*, Graz, 11-13 July 2018. https://eprints.whiterose.ac.uk/133312/

[7] Masdari, M. and Khezri, H. (2020) A Survey and Taxonomy of the Fuzzy Signature-Based Intrusion Detection Systems. *Applied Soft Computing*, **92**, Article ID: 106301. https://doi.org/10.1016/j.asoc.2020.106301

[8] Díaz-Verdejo, J., Muñoz-Calle, J., Estepa Alonso, A., Estepa Alonso, R. and Madinabeitia, G. (2022) On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks. *Applied Sciences*, **12**, Article No. 852. https://doi.org/10.3390/app12020852

[9] Moukhafi, M., Bri, S. and El Yassini, K. (2018) Intrusion Detection System Based on a Behavioral Approach. In: Talbi, E.-G. and Nakib, A., Eds., *Studies in Computational Intelligence*, Springer International Publishing, 61-75. https://doi.org/10.1007/978-3-319-95104-1_4

[10] Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A. and Venkatraman, S. (2019) Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access*, **7**, 41525-41550. https://doi.org/10.1109/access.2019.2895334

[11] Azam, Z., Islam, M.M. and Huda, M.N. (2023) Comparative Analysis of Intrusion Detection Systems and Machine Learning-Based Model Analysis through Decision

Tree. *IEEE Access*, **11**, 80348-80391. https://doi.org/10.1109/access.2023.3296444

[12] Azam, Z., Islam, M.M. and Huda, M.N. (2023) Comparative Analysis of Intrusion Detection Systems and Machine Learning Based Model Analysis through Decision Tree. *IEEE Access*, **11**, 80348-80391.

[13] Mo, X., Pang, J. and Liu, Z. (2024) Deep Autoencoder Architecture with Outliers for Temporal Attributed Network Embedding. *Expert Systems with Applications*, **240**, Article ID: 122596. https://doi.org/10.1016/j.eswa.2023.122596

[14] Yang, Y., Tu, S., Hashim Ali, R., Alasmary, H., Waqas, M. and Nouman Amjad, M. (2023) Intrusion Detection Based on Bidirectional Long Short-Term Memory with Attention Mechanism. *Computers, Materials & Continua*, **74**, 801-815. https://doi.org/10.32604/cmc.2023.031907

[15] Sherstinsky, A. (2020) Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *Physica D*: *Nonlinear Phenomena*, **404**, Article ID: 132306. https://doi.org/10.1016/j.physd.2019.132306

[16] Tokpa, F.W.R., Kamagaté, B.H., Monsan, V. and Oumtanaga, S. (2023) Fake News Detection in Social Media: Hybrid Deep Learning Approaches. *Journal of Advances in Information Technology*, **14**, 606-615. https://doi.org/10.12720/jait.14.3.606-615

[17] Yu, Y., Si, X., Hu, C. and Zhang, J. (2019) A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, **31**, 1235-1270. https://doi.org/10.1162/neco_a_01199

[18] Smagulova, K. and James, A.P. (2019) A Survey on LSTM Memristive Neural Network Architectures and Applications. *The European Physical Journal Special Topics*, **228**, 2313-2324. https://doi.org/10.1140/epjst/e2019-900046-x

[19] Akandeh, A. and Salem, F.M. (2019) Slim LSTM NETWORKS: LSTM_6 and Lstm_C6. 2019 *IEEE* 62*nd International Midwest Symposium on Circuits and Systems* (*MWSCAS*), Dallas, 4-7 August 2019, 630-633. https://doi.org/10.1109/mwscas.2019.8884912

[20] Gill, K.S., Anand, V., Chauhan, R., Choudhary, A. and Gupta, R. (2023) CNN, LSTM, and Bi-LSTM Based Self-Attention Model Classification for User Review Sentiment Analysis. 2023 3*rd International Conference on Smart Generation Computing, Communication and Networking* (*SMART GENCON*), Bangalore, 29-31 December 2023, 1-6. https://doi.org/10.1109/smartgencon60755.2023.10442498

[21] Graves, A., Fernández, S. and Schmidhuber, J. (2005) Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. 15*th International Conference*, *ICANN* 2005, Warsaw, 11-15 September 2005, 799-804. https://doi.org/10.1007/11550907_126

[22] Liu, M., Zhu, T., Ye, J., Meng, Q., Sun, L. and Du, B. (2023) Spatio-Temporal Autoencoder for Traffic Flow Prediction. *IEEE Transactions on Intelligent Transportation Systems*, **24**, 5516-5526. https://doi.org/10.1109/tits.2023.3243913

[23] Sharafaldin, I., Habibi Lashkari, A. and Ghorbani, A.A. (2018) Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *Proceedings of the* 4*th International Conference on Information Systems Security and Privacy*, Funchal, 22-24 January 2018, 108-116. https://doi.org/10.5220/0006639801080116

[24] Intrusion Detection Evaluation Dataset (CIC-IDS2017). https://www.unb.ca/cic/datasets/ids-2017.html