IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 March 2023, accepted 9 April 2023, date of publication 13 April 2023, date of current version 19 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3266979

RESEARCH ARTICLE

A Novel Two-Stage Deep Learning Model for Network Intrusion Detection: LSTM-AE

VANLALRUATA HNAMTE[1], (Graduate Student Member, IEEE), HONG NHUNG-NGUYEN[2], (Graduate Student Member, IEEE), JAMAL HUSSAIN[1], AND YONG HWA-KIM[3], (Member, IEEE)

[1]Department of Mathematics and Computer Science, Mizoram University, Aizawl 796004, India
[2]Department of Information Technology, Viet Tri University of Industry, Viet Tri 29000, Vietnam
[3]Department of Data Science, Korea National University of Transportation, Uiwang, Gyeonggi 16106, Republic of Korea

Corresponding author: Yong Hwa-Kim (yongkim@ut.ac.kr)

This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy (MOTIE) of the Republic of Korea under Grant 20206910100020 and Grant 20221A10100011.

ABSTRACT Machine learning and deep learning techniques are widely used to evaluate intrusion detection systems (IDS) capable of rapidly and automatically recognizing and classifying cyber-attacks on networks and hosts. However, when destructive attacks are becoming more extensive, more challenges develop, needing a comprehensive response. Numerous intrusion detection datasets are publicly accessible for further analysis by the cybersecurity research community. However, no previous research has examined the performance of the proposed model on a variety of publicly accessible datasets in detail. Due to the dynamic nature of the attack and its rapidly changing attack techniques, the publicly accessible intrusion datasets must be updated and benchmarked regularly. The deep neural network (DNN) and convolutional neural network (CNN) are examined in this article as types of deep learning models for developing a flexible and effective IDS capable of detecting and comparing them with the proposed model in detecting cyber-attacks. The constant development of network behavior and the fast growth of attacks need the development of IDS and the evaluation of many datasets produced over time through static and dynamic methods. This kind of research enables the identification of the most efficient algorithm for identifying future cyber-attacks. We proposed a novel two-stage deep learning technique hybridizing Long-Short Term Memory (LSTM) and Auto-Encoders (AE) for detecting attacks. The CICIDS2017 and CSE-CICDIS2018 datasets are used to determine the optimum network parameters for the proposed LSTM-AE. The experimental results show that the proposed hybrid model works well and is applicable for detecting attacks in modern scenarios.

INDEX TERMS Convolutional neural network, deep neural networks, network intrusion detection, deep learning, two-stage model, LSTM-AE.

I. INTRODUCTION

Information and communications technology (ICT) systems and networks manage various sensitive user data that are susceptible to attacks from internal and external attackers. These attacks may be manual or automatic and continuously improve their capabilities, resulting in undetected data breaches. With the growing usage of computer networks in various areas and applications, network security is becoming more essential. Numerous businesses defend themselves against network attacks by using conventional security technologies such as firewalls, anti-spam methods, and anti-virus software. Unfortunately, these technologies are incapable of identifying new or complex threats [1]. As a result, Network Intrusion Detection Systems (NIDS) are now employed as a secondary line of defense to monitor network traffic and identify any intrusive event [2]. A NIDS is a very effective defensive technology capable of mitigating complex attacks and threats [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Elhoseny.

Intrusion detection has always been a significant area of study in network security since it is critical to recognize anomalous access to protected internal networks [4], [5]. A NIDS is gathered by network equipment through mirroring by network devices such as switches, routers, and network terminal access points (TAP). These devices act as a surveillance tool to monitor network infractions and policy breaches [6]. Numerous companies use NIDS in conjunction with firewalls and an application firewall to safeguard web servers on the same network and system. Recently, advanced cyber-attacks circumvent security measures by using irregular patterns like encoding and obfuscation. To address these issues, we used AI-based IDS to identify variant attacks that are undetectable by traditional signature-based NIDS.

Traditionally, intrusion detection relied heavily on conventional methods such as the encryption-decryption method, protocol control, firewalls, and anti-virus software models. While these methods were successful in identifying limited attacks, they had difficulties in detecting a large number of attacks and result in a high rate of false positives. Specifically, hackers launch a huge number of Denial-of-Service (DoS) attacks that are difficult to identify and defend against using traditional methods. The majority of current research has moved its emphasis toward incorporating machine learning (ML) techniques for intrusion detection. In comparison to traditional techniques, they tend to increase identification rates while lowering the overhead associated with managing massive attacks. Support Vector Machines (SVMs) are capable of recognizing target attacks in test datasets based on the properties of the training data and are also memory efficient [7]. The IDS model with SVM uses hyperplanes and kernel functions to identify the attack class. K-nearest neighbor (KNN) design offered a simple, quick, and efficient solution [8]. It classifies the input as samples in the search space and uses a single or multiple feature vector to determine which samples are most similar to the normal or attack class. To minimize the error rate associated with detecting attacks, the naive bayes method uses the probability model with independent assumptions about the characteristics, bias, and variance [9]. The random forests approach [10] ranks features for attack class classification using the integrated feature selection technique and intrinsic metrics. Other methods, like k-means clustering and logistic regression, make use of clustering and regression to determine attack types. Nonetheless, conventional machine learning algorithms have significant shortcomings when it comes to identifying the attacks with highly integrated features [6]. Additionally, these algorithms perform poorly when dealing with noisy and multi-dimensional traffic data. Hybrid machine learning methods combining two or more techniques for IDS have been developed, but they tend to be less efficient due to their large model complexity.

Deep Learning (DL) algorithms have recently advanced intrusion detection [11]. Models for intrusion detection based on non-linear structured deep learning systems such as Deep Neural Networks (DNN) [12], Convolutional Neural Networks (CNN) [13], Recurrent Neural Networks (RNN) [14], and Long-Short Term Memory (LSTM) [15] have shown enhanced learning behaviors and improved accuracy rates for intrusion detection. Additionally, hardware designs have been upgraded in recent years to accommodate the DL models' extensive security enhancements. CNN regularizes the multilayer perceptron and therefore determines the attack class using fully linked networks. Due to the seriousness of this issue, ensuring the security of network systems is imperative. It is critical to use tools to assist with management and security operations. Additionally, these tools must be automated in order to simplify the detection of abnormal occurrences and the implementation of countermeasures to mitigate the impacts of hostile agents. In this study, we propose a two-stage detection system that uses data pre-processing and model training. The model combines an LSTM with an AE to improve detection and accuracy, as illustrated in Figure 1.

In this paper, the main goal of our research is to propose a robust IDS that is able to process a large volume of complicated raw network data efficiently and provide effective detection with higher performance results.

The main contributions of this article can be summarized as follows:

- We present a two-stage Deep Learning-based IDS by hybridizing an LSTM and an AE termed LSTM-AE, where data has been filtered in order to lessen the over-fitting and under-fitting.
- The LSTM-AE can effectively balance the dimensionality reduction and feature retention in highly imbalanced datasets. Therefore, the proposed model has been tested with two datasets.
- The LSTM-AE has a much higher detection performance than other popular intrusion detection models.

To analyze the performance of the LSTM-AE model for IDS, two common error functions, MSE (Mean Squared Error) and MAE (Mean Absolute Error) can be used to compare the performance of the model. MSE measures the average of the squared differences between the predicted and actual values, while MAE measures the average of the absolute differences. In general, MSE is more sensitive to large errors and can be used to penalize the model more heavily for larger errors, while MAE is more robust to outliers and can provide a better estimate of the average error.

To compare the performance of the LSTM-AE model using MSE and MAE, we can train the model on a dataset of network traffic data and evaluate its performance on a test dataset using both error functions. The model that performs better in terms of both error functions is considered to be the better model.

The rest of this article is organized as follows: section I is the introduction to our study. Section II summarizes related works. In section III, we illustrate the preliminaries of deep learning. Section IV presents the proposed model in detail. Section V illustrates the experimental setup used in our study and the result discussion. We evaluate the experimental results and make a comparative analysis with other intrusion detection methods. Finally, section VI concludes the article.

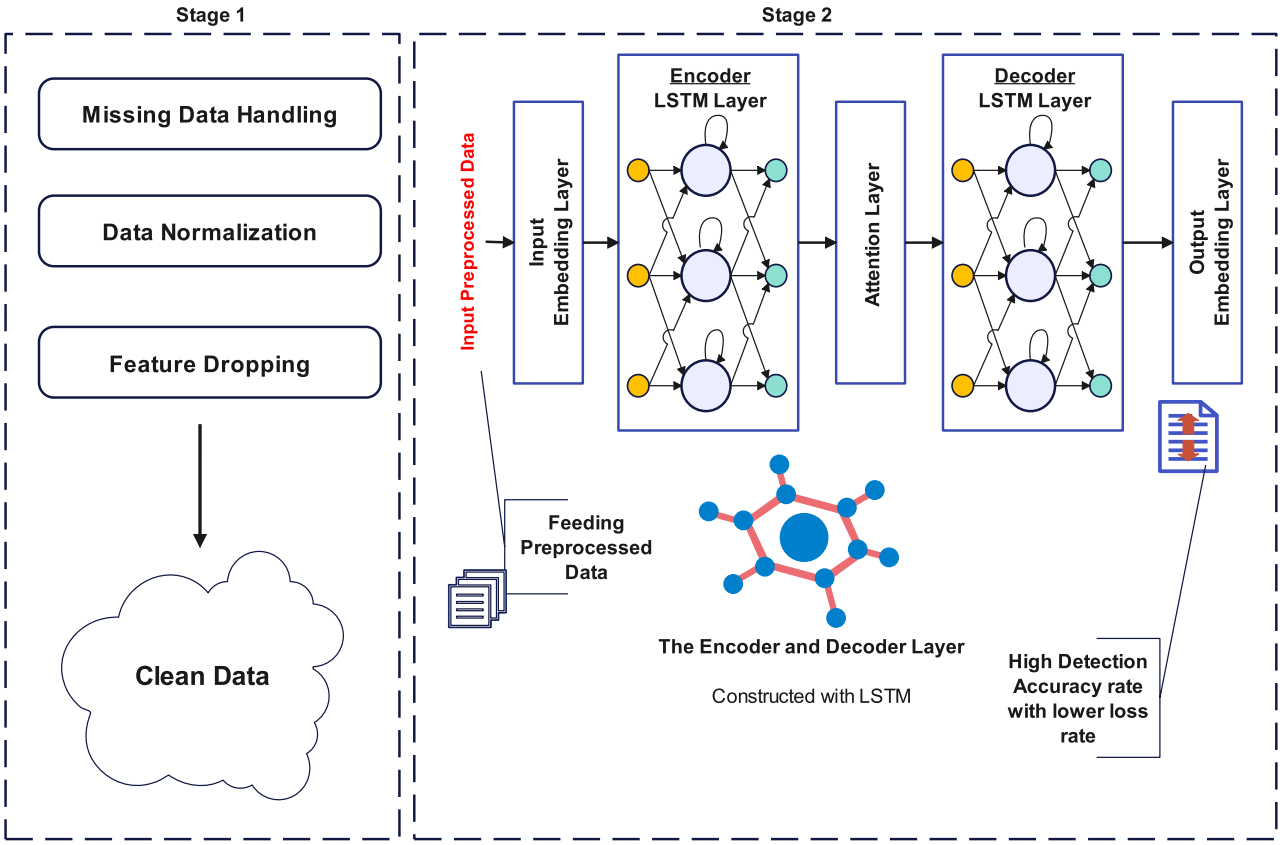


FIGURE 1. Proposed framework—LSTM-AE Model.

II. RELATED WORKS

Since the invention of computer architectures, there has been a study of security problems related to NIDS and HIDS. Since 2017, much research on intrusion detection using a variety of DL methods has been published. The following summarizes relevant research on intrusion detection using deep learning algorithms based on models, features, datasets, and performance measures.

According to Liu et al. [16], intrusion detection models based on convolutional neural networks (CNNs) have the best detection rate and accuracy when compared to other IDS classifiers. Several techniques, models, and approaches based on conventional machine learning have been developed and presented in recent years to address the issue of network intrusion detection. This section is about the current state of machine learning and deep learning techniques applied to the area of network intrusion detection and network intrusion prevention.

Alzahrani and Hong [17] suggest using an Artificial Neural Network with a signature-based approach in the IDS to identify DDoS attacks. After running the tests, it was found that compared to the signature-based method and the use of ANN, the combined strategy was more accurate, obtaining a value of 99.98%.

Kim et al. [18] developed AI-IDS and used a DNN model to evaluate the performance of deep learning models using real-time data and public HTTP datasets from CICIDS 2017, which generated an accuracy rate of 91.69%. When CNN-LSTM was applied to the same dataset, they could obtain a 98.07% accuracy rate. Simultaneously, the paper [18] has highlighted the present IDS detection shortcoming and criticized it for improvement due to its critical nature to security. Additionally, the study suggests using a CNN-LSTM model to conduct payload-level deep learning in a high-performance computing environment.

Zhang et al. [19] used the Monte Carlo tree search method (MCTS) to generate adversarial instances of cross-site scripting (XSS) attacks. Additionally, the authors utilized a GAN framework to enhance the intrusion detection model's ability to identify adversarial attacks. During the experimentation phase, the CICIDS-2017 dataset was used to create novel XSS attacks. Instances of XSS attack traffic and regular traffic were extracted from the dataset. The GAN detection model achieved an accuracy rate of over 99.9% to identify XSS attacks and their adversarial instances.

Srinivas and Manivannan [20] proposed a deep learning method for detecting and blocking DoS-based Hello flood attacks on the medical IoT network. This kind of attack, where numerous Hello packets are delivered to slow down the network, was verified using the Deep Belief Network (DBN) model. In [20], BAU-ROA was used to help the DBN model operate more efficiently and provide better outcomes.

The Rider Optimization Method is a basic optimization algorithm with fewer calculation parameters, while BAU-ROA is a metaheuristic algorithm created to further improve its performance. The tests found that the BAU-ROA method outperforms other optimization algorithms because of its ability to enhance DBN's overall performance.

Ujjan et al. [21] utilized a deep learning model to address the sampling-based technique employed in network security during the early stages of IoT network SDN detection. In their study [21], the SAE system comprises an encoder that reduces the subsequent layer and a decoder that raises the subsequent layer, like a symmetrical system was utilized. To explore the impact of deep learning on intrusion detection, the SAE model was used using adaptive polling and sFlow techniques, and the results were evaluated. Successful findings were achieved for sFlow (a CPU use study with two samples) and Adaptive polling (an accuracy rate study), respectively, as a consequence of the experiment [21].

To identify DDoS attacks, the LSTM deep learning model described in the paper by Priyadarshini and Barik [22] is implemented in the SDN control layer of cloud computing and fog computing environments. Network packets recorded at a particular time interval are most suited for LSTM training since they retain the information of the prior packet's effect on the present packet. The LSTM deep learning model was determined to consist of three hidden layers with 128 hidden units to be adequate as a result of the tests. This level of accuracy was achieved when the LSTM model was applied to the ISCX 2012 and IDS CTU-13 Botnet datasets (when combined, known as Hogzilla). Simulating DDoS attacks is accomplished through the use of open-source tools. Their model achieved an accuracy of 98.88% on the testing dataset.

Krishnan et al. [23] proposed the Deep Autoencoder (NDAE) deep learning model and Random Forest (RF) shallow machine learning method to combat SDN security threats. According to [23], in NDAE, the autoencoder does not have an encoder-decoder structure like that of the conventional autoencoder. In order to address the shortcomings of the shallow machine learning classifiers, researchers favored the usage of deep learning models. NDAE was selected [23] due to its better accuracy and its utilization of less CPU and training time. The DDoS attack detection model employed in NSL-KDD and CICIDS2017 datasets was evaluated via the usage of the NSL-KDD and CICIDS2017 datasets. To test the effectiveness of the model used to identify DDoS attacks, the NSL-KDD and CICIDS2017 datasets were utilized. An evaluation using the NDAE hybrid model found that the model was fit for usage in the intrusion detection system with accuracy rates of 99.60% and 99.24%, respectively.

Kanna and Santhi [24] proposed Optimized CNN (OCNN) and Hierarchical Multi-scale LSTM (HMLSTM) and performed evaluation of publicly available IDS datasets; NSL-KDD, ISCX-IDS, and UNSWNB15. No separate features were needed to combine the technique. An evaluation using the OCNN-HMLSTM model found that the model was fit for usage in the intrusion detection system with an accuracy rate of 90.67%, 95.333%, and 96.334% respectively.

Hussain and Hnamte [25] proposed a DNN model for detecting attacks on the SDN environment. The model was trained with KDD-CUP99, NSL-KDD, and UNSW-NB15 datasets, and achieved 99.61%, 98.12%, and 81.70% accuracy rates respectively. The study had not proposed any pre-processing or data cleaning.

Mighan and Kahani [26] proposed hybridizing Stacked AutoEncoder with Support Vector Machine (SAE-SVM) model to detect an anomaly. The model was trained using ISCX-2012 and CIC-IDS2017 datasets and achieved 95.98% and 99.49% accuracy rates respectively.

Lu and Tian [27] proposed Improved LSTM which is a combination of SAE and Attention-BiLSTM, for Efficient Communication Intrusion Detection. The proposed model was trained with the UNSW-NB15 dataset and achieved a 99.41% accuracy rate in detection.

Binbusayyis and Vaiyapuri [28] suggested an adversarial deep learning method (CNN, GAN, LSTM, and MLP), and the GAN framework performed well in identifying DDoS attacks. The evaluation was performed with a public dataset called CICDDoS 2019 since the dataset contains the most up-to-date types of DDoS attacks.

Wu et al. [29] proposed a Robust Transformer-based Intrusion Detection System (RTIDS), and the model was trained using CICIDS2017 and CIC-DDoS2019 datasets and achieved 99.35% and 98.58% accuracy rate respectively. The study also performed data cleaning before training the models.

Wang et al. [30] proposed a MANifold and Decision boundary-based AE detection system (MANDA) to detect an anomaly. The NSL-KDD and CICIDS-2017 datasets were used to train the model and achieved a 98.41% accuracy rate with a 5% false positive rate. The model performs well, but there was no pre-processing of the data before the training.

Umair et al. [31] proposed a Hybrid Multilayer Deep Learning Model for detecting network intrusion. The model was trained with KDDCUP99 and NSL-KDD datasets and achieved a 99% accuracy rate. Feature selection was also performed before the training phase.

Ravi et al. [32] proposed Recurrent deep learning-based for attack detection and classification. The kernel-based principal component analysis (KPCA) was applied for the feature selection phase, whereas KDD-Cup-1999, UNSW-NB15, WSN-DS, and CICIDS-2017 datasets were used to train the model. The model could achieve a 98% detection accuracy rate trained with the WSN-DS dataset, whereas 99% detection accuracy rate with the rest of the datasets.

Bae and Joe [33] proposed an LSTM-AE based model for UAV anomaly detection with time series data. The proposed model has not been specifically tested for Intrusion Detection Systems, with a focus on the learning time, and no detection accuracy rate has been mentioned. Therefore, the

applicability of their proposed model for IDS is not related to the detection accuracy but only to the usability.

Lindemann et al. [34] studied the LSTM network for anomaly detection, highlighting twelve sources between 2015 to 2020 where LSTM, encoder-decoder-based, and hybrid approaches were proposed. These sources compared the LSTM network's performance with other ML and DL models, but they did not focus on IDS specifically.

Musleh et al. [35] proposed Automatic Generation Control (AGC) based on LSTM with Stacked Autoencoders to maintain the stability and operation of power grids. Although the techniques are model-free and do not require exact system models, there are still numerous obstacles to overcome, such as the high dimensionality of observations that may be difficult to fit. Additionally, the restricted attack scenarios employed during training raise questions about the ability of these detection algorithms to recognize zero-day attacks.

Mushtaq et al. [36] proposed a two-stage IDS by hybridizing the Auto-Encoder and the LSTM. The proposed model achieved 89% accuracy in the classification of attacks, using the NSL-KDD dataset for training.

Mahmoud et al. [37] proposed an AE-LSTM based model to detect anomalies in an IoT environment. The model was trained with the NSL-KDD dataset and achieved 98.88% accuracy in the classification of attacks. The study did not include any data pre-processing to improve training performance.

Altunay and Albayrak [38] proposed a hybrid CNN+ LSTM-based IDS. The UNSW-NB15 dataset was used to train the proposed model, which achieved a 93.21% accuracy rate in binary classification and a 92.9% accuracy rate for multi-class classification. The proposed system was highlighted to be used in an industrial IoT network. Despite the high accuracy detection rate, the loss of training, validation, and testing is very high. No performance boosting was proposed.

Issa and Albayrak [39] proposed DDoS detection by hybridizing CNN and LSTM. The NSL-KDD dataset was used to train the model, which achieved a 99.20% accuracy rate. The dataset is old and may not reflect modern attacks scenerio.

Only a few research studies on intrusion detection have shown that deep learning successfully outperforms conventional techniques. Unsupervised feature learning techniques and algorithms for network intrusion detection include Deep Belief Networks (DBNs), DNNs, Restricted Boltzmann Machines (RBMs), and auto-encoders (AE), among others.

III. PRELIMINARIES OF THE DEEP LEARNING

Deep Learning (DL) is a branch of machine learning that focuses on the function of neurons in the brain. These algorithms are implemented as artificial neural networks (ANNs). A machine learning (ML) or DL algorithm can be trained using various techniques, including supervised and unsupervised learning. Supervised learning involves the classification of data instances that have been labeled during the training phase.

A. DEEP NEURAL NETWORKS

DNNs are generally feedforward networks, in which data flows continuously from the input layer to the output layer, and the connections between the layers are one-way in a forward direction. A fundamental component of a DNN is an ANN inspired by biological neurons found in the human brain. A DNN calculates and transmits the sum of the data received on its input side. Each hidden layer applies an activation function before producing an output, which is essential to facilitate learning and approximation due to the non-linearity of real-world issues. The softmax activation function, given in equation 4, is used in the output layer. The binary cross-entropy loss function is utilized to measure the loss of a sample by calculating it using equation 1.

$$Loss = -\frac{1}{outputsize} \sum_{i=1}^{outputsize} y_i * log\hat{y}_i + (1 - y_i) * log(1 - \hat{y}_i) \quad (1)$$

where $\hat{y}_i$ is the i^{th} scalar value in the model output, y_i is the corresponding target value, and *outputsize* is the number of scalar values in the model output.

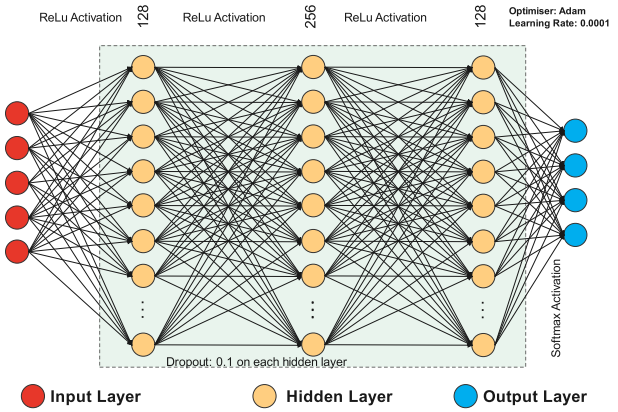


FIGURE 2. DNN model architecture.

In a DNN model, there are no feedback links. The three major components of the Feed-forward Neural Network are the input and output layers, as well as the hidden layer, which may include many hidden units. The layers are each weighed separately. These units are responsible for initiating the activation operations of the units from the previous layer [40].

A simple DNN can be depicted as Figure 2. Each hidden layer with a relu activation function can be as the following mathematical equation:

$$g(x) = f(x^T w + b) \quad (2)$$

$$g(x) = g_i(g_{i-1})(\ldots(g_1(x))) \quad (3)$$

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad (4)$$

$$ReLu = max(0, x) \quad (5)$$

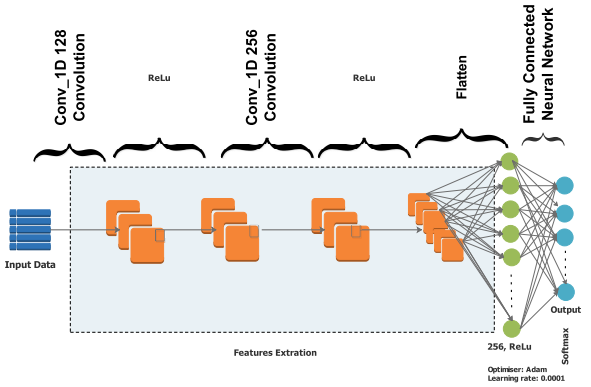


FIGURE 3. CNN model architecture.

B. CONVOLUTIONAL NEURAL NETWORKS

CNNs are a subclass of deep neural networks that were formerly used to analyze visual images, video comprehension, voice recognition, and natural language processing.

By obtaining the features of the kernel function, this convolution function preserves the spatial link between the input data. The kernel values are adjusted automatically based on the optimal structural arrangement. The aspect plot's magnitude is proportional to the layer depth. The supplementary nonlinear function is employed before convolution to generate feature maps. The nonlinear activation may take the shape of a ReLU as formulated in equation 5. The Fully-connected Layer (FCL) is a multi-level neural layer that initiates the result layer via softmax initiation. The previous layer nodes are connected to the next layer nodes. A simple CNN model can be constructed as Figure 3. The categorical cross-entropy loss function is utilized to measure the loss of a sample by calculating the equation 6.

$$Loss = -\sum_{i=1}^{output\_size} y_i * log_i \tag{6}$$

Since the softmax is used to activate the output layer, it is possible to determine the loss function's derivative with respect to each weight in the network and for each value in the training set with the given formula as in equation 6. For other hidden layer activation, relu activation function is used.

IV. METHODOLOGY

The theoretical idea underlying the suggested paradigm is presented in this section. A novel two-stage deep learning model, the LSTM-AE, for network intrusion detection is introduced, and further details on the same topic are provided. Then, we provide a brief overview of the two models that were used to test the proposed model. We will describe our approach before diving into the experiments and discussion.

A. DATASETS

The current and common IDS models utilized a single intrusion detection dataset to concentrate on numerous attacks. On various datasets, not all ideal IDS models performed similarly. The majority of IDS models are optimized for a subset of datasets while performing poorly on others. Thus, well-known public datasets for intrusion detection will be utilized to evaluate the proposed CNN and DNN in this study. The primary goal is to develop an adaptive intrusion detection system (IDS) that performs much better in the majority of datasets with a variety of attacks. Additionally, it is critical to choose suitable datasets, since they play a critical role in the IDS's assessment process. The CICIDS-2017[1] and CSE-CICIDS-2018[2] were chosen as the datasets for this study as IDS Datasets suggested by Hnamte and Hussain [41].

While previous research has mostly utilized KDD99 or KDD, as well as NSL-KDD, these datasets are unsuitable for real-time detection [42]. These datasets are concerned with metadata, which makes it difficult to detect illegitimate attacks in a real workplace context since metadata are not attack attempts. Additionally, the majority of publicly available datasets include duplicate data and an uneven number of categories. Ring et al. [43] examined the features of prior intrusion detection datasets. This research demonstrates that a variety of previously released datasets accurately represent repetitive and unproductive attacks such as DDoS, UDP Flooding, and brute force, which vary from current web-attack trends. Indeed, since attack types and data patterns are continuously changing, it is essential to create a general-purpose model that reflects the current trends [41].

B. DATA PREPROCESSING DATASET

For several years now, AI has attracted considerable scientific attention for developing Anomaly Intrusion Detection System (AIDS) to create a secure network that is resistant to current kinds of attacks, which continues to be a top priority for researchers. Numerous research aims to improve AIDS and assess their progress using a variety of measures. The majority of machine learning algorithms are parameterized, which means that their behavior cannot be predicted only on the basis of the processed data. Additionally, random factors have a major effect on how well AIDS models work. Following that, the behavior of parameters must be fine-tuned to get an acceptable evaluation.

Data cleaning is the process of identifying and correcting or removing errors, inconsistencies, and discrepancies in data before it is used for analysis or modeling. It is an essential step in data preprocessing, especially for deep learning applications. Here are some reasons why data cleaning is necessary for deep learning:

- Garbage in, garbage out: The quality of the data used to train a deep learning model has a significant impact on the quality of the model's predictions. If the data is noisy, inconsistent, or contains errors, the model's predictions will likely be inaccurate or unreliable.
- Overfitting: Deep learning models are prone to overfitting, which occurs when the model memorizes the

[1]CICIDS-2017 https://www.unb.ca/cic/datasets/ids-2017.html
[2]CSE-CICIDS-2018 https://www.unb.ca/cic/datasets/ids-2018.html

training data instead of learning the underlying patterns. If the training data is noisy or contains errors, the model may overfit to these errors, resulting in poor generalization to new data.

- Optimization: Deep learning models are optimized using gradient-based optimization algorithms, which require the data to be smooth and continuous. If the data contains outliers or other anomalies, the optimization algorithm may struggle to find the global minimum.
- Efficiency: Deep learning models require large amounts of data to train, and cleaning the data can help reduce the amount of data required. Removing irrelevant or redundant features, for example, can reduce the dimensionality of the data and make it more efficient to train.
- Ethics and fairness: Data cleaning can also help ensure that the data used to train the model is representative and unbiased. Biases in the data can lead to unfair or unethical predictions, such as racial or gender biases.

To avoid differences in the outcomes of feature quantification and to prevent features with a wide value range from impacting the model's output, the datasets are normalized for all features. Normalization of data may enhance the model's precision and accelerate the model solution. The Min-Max normalization method is applied to assume that the dataset of a group of features is $X_1, X_2, \ldots X_n$ the equation for normalizing a certain data X_i in the set is shown in the following equation:

$$X_i' = \frac{X_i - X_{Min}}{X_{Max} - X_{Min}} \tag{7}$$

Here, X_i' is the normalized data and $X_M in$ and $X_M ax$ are the minimum and maximum values in the dataset, respectively.

Furthermore, it is worth noting that in CSE-CICIDS2018, there are only 928 Web Attacks available, while the sample size of the dataset has significantly increased compared to the CICIDS-2017 IDS dataset, particularly for Botnet and Infiltration attacks, which have risen by 143 and 4497, respectively. The comparison between the sample sizes of both datasets is presented in Table 1.

For optimal results, the data must be refined and standardized before training. There are null, redundant, and outlier values in the datasets. Similarly, dataset samples do not all fall within the same range and must be normalized before training for effective prediction. From the datasets, null, redundant, and outlier values are removed. We used the standard transform for data normalization since it centers and scales each feature separately, following every feasible examination of null and redundant data removal techniques. Table 2 has shown the features removed from the CICIDS-2017 dataset, whereas Table 3 has shown the features removed from the CSE-CICIDS-2018 dataset.

In summary, data cleaning is essential for deep learning applications to ensure the quality, reliability, and fairness of the model's predictions. Data cleaning can help prevent overfitting, improve optimization, increase efficiency, and promote ethical and fair use of the model.

C. PROPOSED FRAMEWORK

Traditional approaches have the drawbacks of starting from scratch, overfitting, and short-term memory issues when the amount of data increases or the correlation between variables becomes complex. Using sequential learning models, these problems are relatively solvable. However, simulating the temporal and spatial characteristics of network attacks is complex. In this paper, we propose a novel two-stage model that combines AE and LSTM with a data preprocessing phase to effectively identify attack categories from regular traffic. The architecture of the proposed model is shown in Figure 6. The AE encodes the original data and forms a bottleneck, and the decoding network restores all the data. The main challenges of the proposed model are the combination of two types of architectures and the training with smoothing constraints. The bottleneck is a normal layer whose main goal is to recover current data, as shown in Figure 4.

The AE is an unsupervised algorithm that sets the target values to be equal to its input signals. The AE tries to learn a function $h_{w,b}(X)$ that satisfies $h_{w,b}(X) \approx X$, where $X = x_1, x_2, \ldots x_n$ represents the original signals, and w and b are hyperparameters. This implies that the AE tries to learn an approximation to the identification function so that output values similar to X can be obtained. Usually, an AE can be divided into encode/decode layers. The encoding layers compress the original signals to obtain a low-dimensional representation, which corresponds to the so-called bottleneck features. The decoding layers reconstruct the inputs from the bottleneck features. Thus, the objective function of the AE can be written as equation 8.

$$L(h_{w,b}, X) = \frac{1}{2} \parallel h_{w,b}(X) - X \parallel^2 \tag{8}$$

The forward calculation of the $\ell + 1$ layers in AE is written as equation 9.

$$\alpha^{(\ell+1)} = f(W^\ell \alpha^\ell + b^\ell) \tag{9}$$

where f is the activation function and $W^\ell \alpha^\ell$ and b^ℓ are the weights, activation value, and bias of the ℓ layer, respectively. We first compute the forward output value of the hypothesis $h_{w,b}, X$. Then, for each node i in layer ℓ, we compute an "error term" δ_i^ℓ; it measures how much this unit was a "response" for the output error. For the units in the output layer, the difference between network's activation and target value is intuitive. It can be represented as equation 10.

$$\delta_i^{n_\ell} = \frac{\partial L(h_{w,b}, X)}{\partial z_i^{n_\ell}} = -(X^i - a_i^{n_\ell}) f'(z_i^{n_\ell}) \tag{10}$$

where n_ℓ denotes the output layer, $z_i^{n_\ell}$ and $a_i^{n_\ell}$ denote the total weighted sum of inputs and the activation to unit i in layer n_ℓ, respectively, and f is the activation function. For the other layers ℓ, the equation is defined as equation 11.

$$\delta_i^\ell = \left(\sum_{j=1}^{s_{\ell+1}} W_{ij}^\ell \partial_j^{\ell+1} \right) f'(z_i^\ell) \tag{11}$$

TABLE 1. Content of CSE-CICIDS-2018 dataset and CICIDS-2017 dataset.

Dataset	Normal	DDoS	Dos	Botnet	Brute Force	Infiltration	Web Attacks	Port Scan
CICIDS-2017	1743179	128027	252661	1966	13835	36	2180	158930
CSE-CICIDS2018	6112151	687742	654301	286191	380949	161934	928	-

TABLE 2. CIC-IDS2017 dataset dropped features.

Features	Description
Flow ID	Unique Flow Identification allotted
Source IP	The source IP Address
Destination IP	The destination IP Address
Timestamp	The accessing date and time

TABLE 3. CSE-CICIDS-2018 dataset dropped features.

Features	Description
Protocol	The Internet Protocol
Destination Port	The destination Port
Timestamp	The accessing date and time

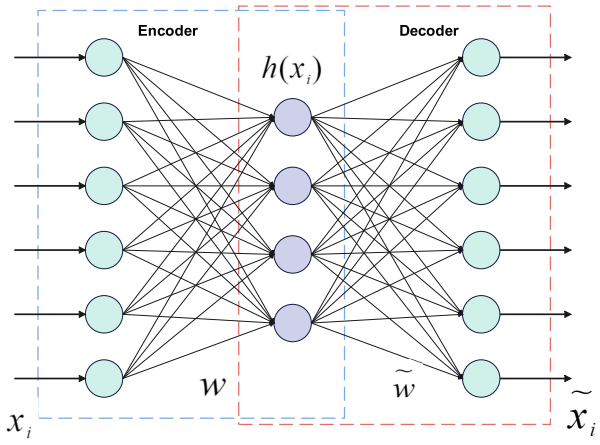


FIGURE 4. AE model architecture.

and the update rules of W, b are defined as equation 12 and equation 13.

$$W_i^{\ell} = W_i^{\ell} - \alpha a_j^{\ell} \partial_i^{\ell+1} \tag{12}$$

$$b_i^{\ell} = b_i^{\ell} - \alpha \partial_i^{\ell+1} \tag{13}$$

where α is the iteration step length.

The primary purpose of the AE is to minimize the data dimensions of inputs while preserving the essential data structure information. The most often used forms for encoders and decoders are mathematical transformations that maintain collinearity (Equation 14) and nonlinearity (Equation 15):

$$f_\theta(x_i) = s_f(b + W) \tag{14}$$

$$g_\theta(h(x_i)) = s_g(c + \widetilde{W}) \tag{15}$$

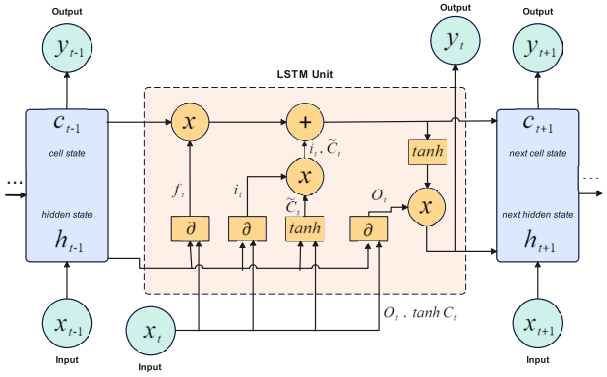


FIGURE 5. LSTM model architecture.

whereas s_f and s_g are the encoder and the decoder activation function, such as sigmoid, etc., b and c are the bias vectors in the encoder and the decoder layers, w and $\widetilde{w}$ are the weight matrices in the encoder and the decoder.

AE is often employed in representation learning to comprehend unsupervised feature vector inputs. Figure 6 depicts the typical technique with an LSTM within AE which we then term LSTM-AE. Initially, we performed data cleaning as shown in Algorithm 1, then, We applied sequence-to-sequence AE, LSTM network in the encoder and decoder layer as shown in Algorithm 2. The optimum objective is to forecast both short-term and long-term network attacks. AE is composed of an encoder and a decoder, with the input sequence first being encoded and then decoded.

Another prominent deep learning model is the recurrent neural network (RNN), where connections between units create a directed graph coupled with sequence information from the input. The RNN analyses a series of input data by using their internal state, which results in a vanishing gradient issue that has a significant severe impact on model accuracy [44]. LSTM is an improved form of RNN that avoids the vanishing gradient issue by the use of gates (input, forget, and output) and memory cells as depicted in Figure 5.

The LSTM sequence is a list of LSTM units to capture the temporal information. The encoder and decoder consist of the LSTM sequences. After encoding, the bottleneck features are fed into the decoder for data reconstruction. The residual between the recovery and input data for adjusting the network is computed by square error. Bidirectional architecture has not been taken considered in the proposed model. The following equations show the LSTM operation.

$$f_t = \Phi(\hat{W}_f.[h_{t-1}, x_t] + B_f) \tag{16}$$

$$i_t = \Phi(\hat{W}_i.[h_{t-1}, x_t] + B_i) \tag{17}$$

$$\widetilde{C}_t = \tanh(\hat{W}_C.[h_{t-1}, x_t] + B_C) \tag{18}$$

$$C_t = f_t \times C_{t-1} = i_t \times C_t \tag{19}$$

$$O_t = \Phi(\hat{W}_O.[h_{t-1}, x_t] + B_O) \tag{20}$$

$$h_t = O_t \times \tanh(\Phi(C_t)) \tag{21}$$

As shown in Figure 5, for the equation 16, x_t is the network input, h_t is the output from the hidden layer, Φ denotes the sigmoid function, C_t is the cell state and the state candidate values are denoted using $\widetilde{C}_t$, $\hat{W}_i$, $\hat{W}_O$, $\hat{W}_f$ and $\hat{W}_c$ are the weight used on the input, drop and output gate and memory cell, whereas B_f, B_i and B_C denote the bias for the input and output as well as the forget, gate and cell which are used in equation 16, 17, 18, 19, 20 and 21. The input gate determines if input data will be retained, the forget gate determines whether data will be lost, the cell records the processing status, and the output gate delivers the result. This architecture was built specifically to overcome the gradient issue in RNNs.

Algorithm 1 Stage 1: Data Processing

Input: $D = [X_1, X_2, \ldots X_n]$
Output: Cleaned Dataset

Set $D_{Length} = Row\ length$ from D
for $i \leftarrow 1$ to $i <= D_{Length}$ **do**
  if $D_{(X_1, X_2, \ldots X_n)}$ is of **NULL** *or* **na** type **then**
    Remove Row: D_i
  end if
end for
Update D
Apply Data Normalization to D
Preprocessed MinMaxScaler (Eqn 7) to D
Output D

Algorithm 2 Stage 2: LSTM-AE

Input: $I_d = [X_1, X_2, \ldots X_n]$
Output: Intrusion detection accuracy

Split Data into Training and Testing Set:
$X_{Train} = X_1, X_2, \ldots X_(n-t)$
$X_{Test} = X_(n-t+1), \ldots X_(n)$
Train Model:
$AE \leftarrow$ AutoEncoder (X_{Train})
EncoderModel (FeedForwardAE(AE)): Eqn 9
for each X_j in X_{Train} **do**
  Calculate f_t (Eqn 16), $\widetilde{C}_t$ (Eqn 18, i_t (Eqn 17)
  Update Cell State c_t (Eqn 19)
  Calculate O_t (Eqn 20), h_t (Eqn 21)
end for
$AE \rightarrow$ output: Eqn 10
Calculate δ_i^ℓ (Eqn 11), W_i^ℓ (Eqn 12), b_i^ℓ (Eqn 13)
Apply MAE (Eqn 23) for the Error Rate

In this study, encoders based on unsupervised training are employed to reinstate the latent vectors to the input sequences, driving the encoders to learn how to create more meaningful latent vectors, hence mitigating the issue of feature loss and local optima in the training process. The two auxiliary decoders, like the prediction decoder, use the mean square error loss function. The global loss function may then be derived by weighted addition of the loss function, as shown in Equation 22 for MSE and 23 for MAE.

TABLE 4. LSTM-AE: parameters.

Parameter	Value
AE Layer	1
LSTM Layers	2
LSTM Hidden Units	196,125
LSTM Activation	relu
Learning Rate	0.001
Metrics	Accuracy
Loss Function	Mean Absolute Error
Optimizer	Adam
Batch Size	128
Epochs	30

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{22}$$

$$MAE = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n} \tag{23}$$

Table 4 outlines the key hyperparameters for the proposed model, including the number of LSTM layers and hidden units, the activation function and dropout rate for the LSTM layers, the number of AE layer and units, the learning rate, loss function, optimizer, batch size, and number of epochs for training.

V. EXPERIMENTAL AND RESULT DISCUSSION

In order to verify the quality of the LSTM-AE neural network for anomaly detection, two additional models, simple DNN and CNN, were constructed and compared with the LSTM-AE. The structures of these models were set up as follows.

DNN: The neural network was a 4-layer fully connected one with a softmax layer as the output layer. The numbers of neurons in input layer and output layer were both 81. The numbers of neurons in the three hidden layers were 128 and 256 and 128, respectively.

CNN: The convolutional neural network was a 2-layer fully connected to one hidden layer then one with a softmax layer as the output layer. The numbers of neurons in input layer and output layer were both 81. The numbers of neurons in the one hidden layers was 256.

Two publicly accessible datasets were utilized to analyze and verify the proposed LSTM-AE model for NIDS, as well as for DNN and CNN model. These are the CICIDS-2017 and CSE-CICIDS-2018 benchmark datasets.

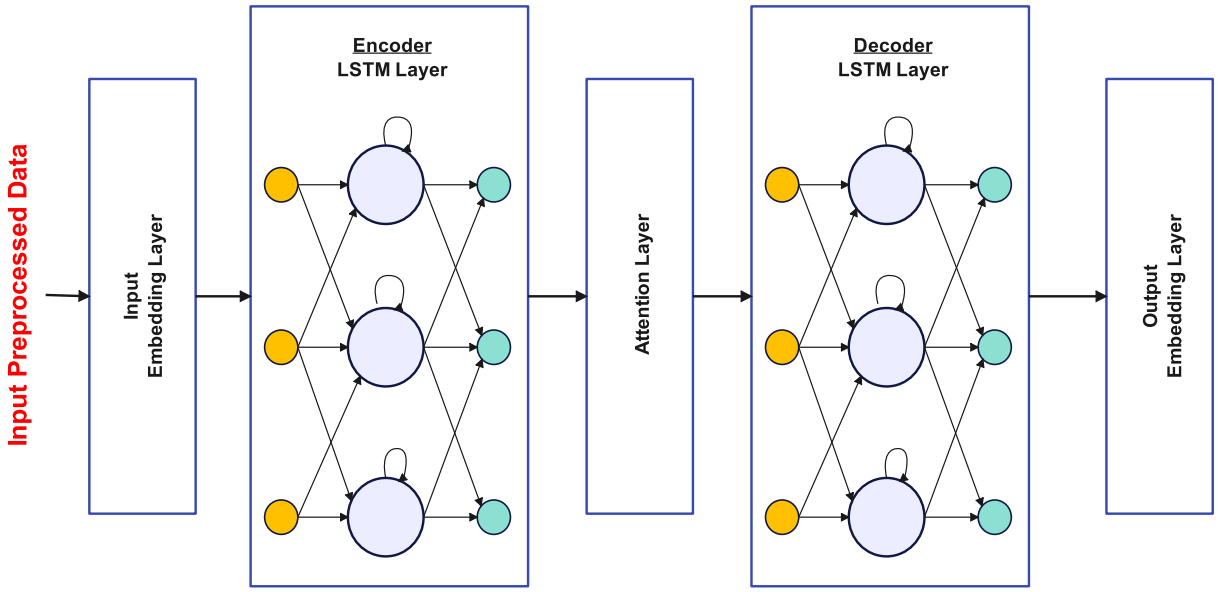


FIGURE 6. LSTM-AE model architecture.

A. CICIDS-2017

The CICIDS-2017 dataset was intended to address the lack of real-time network traffic datasets for intrusion detection evaluation. CICIDS-2017 includes the most current and relevant data for testing security systems. However, the fundamental reason for using this dataset is because it contains information on class imbalances, as mentioned in Panigrahi and Borah [45] and Injadat et al. [46]. Other IDS datasets, such as NSL-KDD [47] or UNSW-NB15 [48], have a limited number of features; for example, NSL-KDD has 42 features [47]; UNSW-NB15 contains 49 features [48]; while CICIDS-2017 contains more than 80 features [49]. As a result, we believe that the CICIDS-2017 dataset is better in terms of data dimension.

CICIDS-2017 is composed of eight (8) sessions of traffic monitoring: Friday-Working Hours-Afternoon-DDos, Friday-Working Hours-Afternoon-PortScan, Friday-Working Hours-Morning, Monday-Working Hours, Thursday-Afternoon-Working Hours-Infiltration, Thursday-Working Hours-Morning-WebAttacks, Tuesday, and Wednesday working hours. Each session is stored in CSV file format. The files include both regular traffic, denoted by the term "BENIGN," and abnormal traffic, also termed "ATTACKS." Apart from regular and benign traffic, this dataset contains 14 other types of attacks. Additionally, the data in the table demonstrates an unbalanced distribution of data among the 15 classes. The percentage of data distributed against the primary class and the distribution of data within each class both illustrate the data's imbalance. The dataset contains classes for a limited number of traffic threats, including Web Attack-SQL Injection, Infiltration, and Heartbleed.

B. CSE-CICIDS-2018

This dataset is the product of a collaborative effort by the Communications Security Establishment (CSE) and The Canadian Institute for Cybersecurity (CIC), which used the concept of profiles to build a comprehensive cybersecurity dataset. It also includes abstract distribution models for various network elements such as applications, protocols, and lower-level components. The dataset contains seven distinct attack scenarios, including Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and network infiltration from inside. The attacking network consists of 50 computers, whereas the victim group's five departments are comprised of 420 PCs and 30 servers. This dataset contains the network traffic and logs files of each victim system, as well as 80 network traffic characteristics collected using CICFlowMeter-V3 from intercepted network traffic [49].

The data set is grouped by day for a total of eight days. Each day, every computer containing the raw data network traffic (Packet Captures) and event logs (Windows and Ubuntu event Logs) were collected. More than 80 traffic characteristics are extracted from the raw data throughout the features extraction procedure which is then available in CSV format.

C. SYSTEM SETUP

Experiments are performed using Python 3.7 with development libraries for the DL applications Keras[3] and TensorFlow,[4] on a desktop computer equipped with an Intel Core i9-10900K CPU, 128 GB of RAM, NVIDIA RTX 3060 Ti GPU with cuDNN installed, and a 64-bit version of Windows 11 Pro.

To evaluate the proposed system's performance, we performed experiments in a variety of situations and compared them to many comparable systems such as DNN, CNN, and LSTM-AE. During the training phase, the following parameters were set: dropout rate of 0.2 and 0.1 to avoid

[3]Keras https://keras.io/
[4]TensorFlow https://www.tensorflow.org/

overfitting; loss function with binary cross-entropy for DNN and categorical cross-entropy for CNN, which is a classic loss function used in binary classification tasks; learning rate of 0.0001; and Adam optimizer, which is an adaptive learning rate optimization algorithm for training deep neural networks.

D. EVALUATION METRIC

To assess the performance of the LSTM-AE model, the same evaluation criteria used in the majority of past research on NIDSs are utilized. In particular, a study of the relevant literature reveals that accuracy, precision, recall, F1-Score, and false alarm rate (FAR) metrics have been frequently utilized to assess the efficacy of the majority of intrusion detection systems [50], [51].

The following formulae are used to calculate these metrics:

$$\text{TPR} = \frac{TP}{TP + FN} \tag{24}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{25}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{26}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{27}$$

$$\text{F1-Score} = 2 * \frac{(\text{Precision} * \text{Recall})}{\text{Precision} + \text{Recall}} \tag{28}$$

$$\text{Accuracy} = \frac{TN + TP}{TN + TP + FN + FP} \tag{29}$$

TP, TN, FP, and FN are abbreviations for True Positive, True Negative, False Positive, and False Negative, respectively. Finally, the F1 Score is the harmonic mean of recall and accuracy, indicating the system's performance appropriately. Equation 28 provides the statistical formulation.

E. OPTIMIZATION AND LAYERS

When the learning rate is reduced from 0.001 to 0.0001, the accuracy of all evaluation measures improves. However, if the learning rate is lowered below 0.0001, all evaluation metrics become ineffective. The loss and accuracy measurements indicate that reducing the learning rate negatively affects the performance of the NIDS model.

For optimization, we use Adam, an adaptive learning rate method that calculates individual learning rates for different parameters. Adam adapts the learning rate for each weight of the neural network by estimating the first and second momentum of a gradient.

We compare the performance of the proposed LSTM-AE model with a CNN model with optimal settings and a DNN model with optimal settings trained on the original CICIDS2017 and CSE-CICIDS2018 datasets, as shown in Figure 7. While ensuring that the false alarm rate does not increase, the LSTM-AE model significantly improves the ability of the model to detect anomalous instances. The loss performance is depicted in Figure 7c and 7d, while Figure 7a and 7b illustrate the accuracy performance for both training and validation.

TABLE 5. Performance comparison of DNN, CNN and proposed LSTM-AE.

CICIDS-2017			
Score	**CNN**	**DNN**	**LSTM-AE**
Accuracy	99.97%	99.93%	**99.99%**
Loss	0.0009	0.0003	0.0005
Recall	99.97%	99.93%	**99.99%**
Precision	99.97%	99.93%	**99.99%**
F-Measure	99.97%	99.93%	**99.99%**
Training Time (epoch)	∼73s	∼55s	**∼184s**
Inference Time	24.65s	28.61s	**53.66s**
CSE-CICIDS-2018			
Score	**CNN**	**DNN**	**LSTM-AE**
Accuracy	98.98%	98.55%	**99.10%**
Loss	0.0245	0.0107	0.0040
Recall	98.97%	98.55%	**99.10%**
Precision	98.94%	98.41%	**99.07%**
F-Measure	98.86%	98.31%	**99.02%**
Training Time (epoch)	∼190s	∼130s	**∼462s**
Inference Time	62.55s	73.14s	**128.24s**

Table 5 shows a performance comparison of three deep learning models: DNN, CNN, and the proposed LSTM-AE on two different datasets, CICIDS-2017 and CSE-CICIDS-2018. The table has two main sections, one for each dataset, each including the scores of the models in terms of accuracy, loss, recall, precision, and F-measure, as well as training and inference times. In each column, the scores of the models are presented, where the rows represent evaluation metrics, and the columns represent the three models. Each metric is evaluated using a percentage or a time value. In each metric, the value in bold represents the best performance among the three models. For example, in the CICIDS-2017 section, the LSTM-AE model has the best performance in terms of accuracy, loss, recall, precision, and F-measure. However, the CNN and DNN models have better training and inference times. Similarly, in the CSE-CICIDS-2018 section, the LSTM-AE model outperforms the CNN and DNN models in terms of accuracy, recall, precision, and F-measure, but the CNN and DNN models have better training and inference times.

Additionally, Table 5 demonstrates that the training time for DL models can vary significantly depending on various factors such as the complexity of the model, the size of the dataset, and the optimization techniques employed. As shown in Figure 8, for smaller datasets and simpler models, training can take only a few seconds when our proposed model is trained on GPU. To optimize training time, various techniques such as mini-batch training, early stopping, and transfer learning can be employed. Additionally, pre-trained models can also be used as a starting point for further training, reducing the time required for training.

When training the proposed LSTM-AE model, there are several advantages to training it on a GPU (Graphics Processing Unit) rather than a CPU (Central Processing Unit):

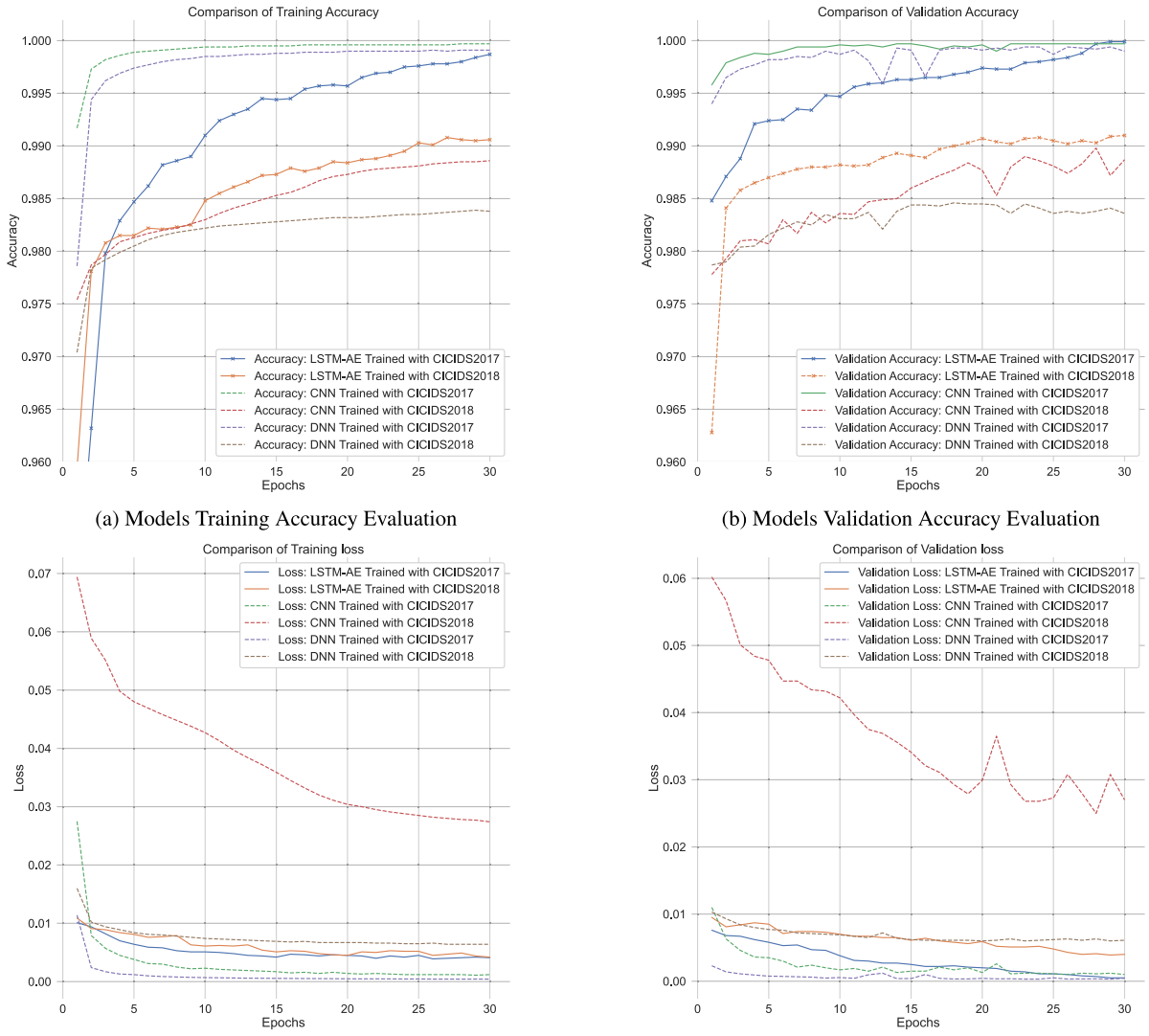
- Speed: One of the main advantages of training an LSTM-AE model on a GPU is that GPUs are specifically designed to perform the types of calculations required

for deep learning models. As a result, training the proposed LSTM-AE model on a GPU was much faster. This is because GPUs have a large number of cores that can perform calculations in parallel, which can significantly reduce the training time.
- Memory: Another advantage of training the proposed LSTM-AE model on a GPU was that GPUs have more memory than CPUs. This can be particularly true when working with large datasets, as in our study, and as it allowed the model to process more data at once without running out of memory.

Here are some reasons why the proposed LSTM-AE may be better suited for intrusion detection tasks compared to DNN and CNN:
- Sequential data processing: LSTM-AE is well-suited for processing sequential data, such as time series or natural language data, which require a model to remember and analyze past events or words. LSTM-AE's memory cells allow it to retain information for an extended period, making it useful for tasks that require a model to remember patterns over time.
- Feature extraction: Autoencoders, including LSTM-AE, are well-suited for feature extraction tasks, where the goal is to identify and represent the most important features of the input data. This makes LSTM-AE useful for dimensionality reduction tasks, where it can learn a compressed representation of the input data.
- Anomaly detection: LSTM-AE is useful for anomaly detection tasks, where the goal is to identify unusual or unexpected patterns in the data. Because LSTM-AE is trained to reconstruct the input data, it can identify anomalies that do not fit the learned patterns.
- Data efficiency: LSTM-AE can be more data-efficient than DNN and CNN, especially in cases where there is imbalanced training data available. This is because LSTM-AE can learn to generalize from a set of examples by capturing the underlying patterns in the data.

Table 6 compares the performance of different models used for IDS in terms of their accuracy. The proposed LSTM-AE model was assessed and compared to that of DNN and CNN baseline models, which were also employed for the same dataset. The table lists the year of publication, reference, algorithm, number of features used, dataset used, and accuracy achieved by each model. The proposed LSTM-AE is compared with other models and is found to have a better accuracy than some models but not all. The accuracy of different models ranges from 65.1% to 99.90%. The table highlights the diverse range of algorithms and features used in IDS research. However, some studies have used datasets that are a decade or more old for training and validation, which may not accurately reflect modern attack scenarios. It is important to acknowledge that each dataset listed in table 6 has its own unique characteristics and limitations, and it would be inappropriate to criticize all of them in the same way. Nonetheless, there are some potential criticisms of each dataset that are worth mentioning:
- KDDCUP99: This dataset is often criticized for being outdated and not representative of modern network traffic. It also contains a lot of redundant and irrelevant features, which can make it difficult to build accurate models.
- DARPA1998: This dataset is also quite old and may not accurately represent modern network traffic. It is also relatively small, which can limit the complexity of models that can be trained on it.
- ISCX2012: This dataset is more recent than the previous two and contains a wider variety of network traffic, but it is still relatively small and may not be fully representative of all types of network traffic.
- UNSW-NB15: This dataset is relatively recent and contains a wide variety of network traffic, but it has been criticized for containing a lot of noisy data and being difficult to work with.
- WSN-DS: It is difficult to provide a general criticism of this dataset without more information about it. However, wireless sensor network datasets in general can be difficult to work with due to the complex, heterogeneous nature of the data they produce.

As shown in Table 6, some studies rely on a single dataset to validate their proposed model. However, such an approach is not ideal as it fails to ensure the generalization of the model. This is because a single dataset can have unique characteristics that the model might learn and overfit to, leading to a high accuracy score for that dataset but low performance on other datasets. To test the generalizability of a model, it is crucial to evaluate it on multiple datasets that have different characteristics. A model that performs well across various datasets demonstrates its suitability for different environments, which is a crucial requirement for an effective IDS. Validating a proposed model using two or more datasets can have several advantages:
- **Generalizability**: The model that performs well on multiple datasets can be considered to be more generalizable, meaning that it can perform well on datasets that it has not seen before. This can be especially important when the model is being deployed in a real-world setting where it will encounter a range of data it has not seen before.
- **Robustness**: Testing a model on multiple datasets can help to identify any weaknesses or flaws in the model. A model that performs well on multiple datasets is likely to be more robust than one that only performs well on a single dataset.
- **Reduction of overfitting**: Overfitting occurs when a model is too closely fitted to the training data and does not generalize well to new data. Validating a model on multiple datasets can help to identify overfitting, and can help to ensure that the model is not overly specialized to the training data.
- **Increased credibility**: Validating a model on multiple datasets can increase the credibility of the results, as it shows that the performance of the model is not



(a) Models Training Accuracy Evaluation

(b) Models Validation Accuracy Evaluation

(c) Models Training Loss Rate Evaluation

(d) Models Validation Loss Rate Evaluation

FIGURE 7. **Model performance comparison.**

due to chance or specific characteristics of a particular dataset.

- **Comparison with other models**: Validating a proposed model on multiple datasets can enable direct comparison with other existing models. This can help to identify strengths and weaknesses of the proposed model and provide insights for further improvement.

The findings for accuracy, loss, recall, precision, and F-measure were compared, and the overall results are depicted in Figure 7. The proposed model was trained using MAE as the error function, which resulted in the lowest loss error rates. Therefore, the proposed method was able to achieve the smallest error rate among these models, as shown in Table 5. We compared our proposed LSTM-AE model with other studies, as shown in Table 6.

We compared the performance of the model using MSE as the error function with two different learning rates, 0.001 and 0.0001, as presented in Table 7. While using MSE as the error function can effectively train the model to minimize the differences between predicted and actual values up to 5 epochs, it may not always be the optimal choice for all types of models or datasets. Therefore, we also applied MAE as the error function with learning rates of 0.001 and 0.0001, as shown in Table 8. It is important to consider different error functions and metrics to evaluate the performance of a model comprehensively.

MAE and MSE are both commonly used error functions in machine learning, but they differ in the way they measure the errors between the predicted and actual values. The main advantage of using MAE over MSE is that MAE is more robust to outliers, while MSE penalizes outliers more heavily, which is clearly visible from comparing Table 7 and Table 8.

One advantage of using MAE (equation 23) over MSE (equation 22) is that MAE is more resistant to outliers, which

TABLE 6. Performance comparison of the proposed LSTM-AE with other models.

Sl. No.	Year	Reference	Algorithm	Features	Dataset	Accuracy
1	2018	[17]	ANN	*	Self Generated	99.95%
2	2018	[52]	HAST-IDS	40, 19	DARPA1998, ISCX 2012	99.68% 99.89%
3	2019	[53]	SMOTE + PCA	80	CICIDS2017	81.83%
4	2019	[54]	UDDB + AE	80	CICIDS2017	99.60%
5	2019	[40]	DNN	41, 41, 41, 23, 80	KDDCUP99, NSL-KDD, UNSW-NB15, WSN-DS, CICIDS2017	92.5% 78.5% 65.1% 96.4% 95.6%
6	2019	[55]	CNN	41	NSL-KDD	89.41%
7	2019	[56]	RNN	80, 41	CICIDS2017, NSL-KDD	99.10% 99.40%
8	2019	[23]	VARMAN	80 41	CICIDS2017, NSL-KDD	99.24% 99.60%
9	2019	[22]	LSTM 2.2	192	Hogzilla	98.88%
10	2020	[18]	CNN-LSTM	80	CICIDS2017	98.07%
11	2020	[19]	LSTM-D	80	CICIDS2017	99.90%
12	2020	[57]	IFAN-IDS	80	CICIDS2017	84.45%
13	2020	[58]	ANN	80	CICIDS2017	96.24%
14	2020	[36]	AE-LSTM	41	NSL-KDD	89%
15	2021	[24]	OCNN-HMLSTM	41, 19, 49	NSL-KDD, ISCX-IDS, UNSW-NB15	90.67% 95.33% 96.33%
16	2021	[25]	DNN	41, 41, 49	KDDCUP99, NSL-KDD, UNSW-NB15	99.61% 98.12% 81.70%
17	2021	[26]	SAE-SVM	19, 80	ISCX-2012, CIC-IDS2017	95.98% 99.48%
18	2021	[27]	SAE+Attention-BiLSTM	49	UNSW-NB15	99.41%
19	2021	[28]	CAE-OCSVM	41, 49	NSL-KDD, UNSW-NB15	91.58% 94.28%
20	2022	[29]	RTIDS	81, 87	CIC-IDS2017, CIC-DDoS2019	99.35% 98.58%
21	2022	[30]	MANDA	41, 81	NSL-KDD, CIC-IDS2017	98.41%
22	2022	[31]	Hybrid Multilayer DL	41, 41	NSL-KDD, KDDCUP99	99%
23	2022	[32]	Recurrent DL	41, 49, 19, 81	KDDCUP99, UNSW-NB15, WSN-DS, CICIDS-2017	99% 99% 98% 99%
24	2023	[37]	AE-LSTM	41	NSL-KDD	98.88%
25	2023	[38]	AE-LSTM	49	UNSW-NB15	93.21%, 92.9%
26	2023	[39]	CNN+LSTM	41	NSL-KDD	99.20%
27	***This paper***	**	LSTM-AE	81 76	CICIDS2017 CSE-CICIDS2018	**99.99%** **99.10%**

* The Features are not mentioned or not available.
** This paper.

TABLE 7. Performance comparison LSTM-AE with MSE error function.

With 5 epochs training Loss Function = MSE		Learning Rate	
Dataset	**Performance**	**0.0001**	**0.001**
CICIDS2017	Accuracy	93.98%	92.48%
	Loss	0.0040	0.0030
	∼Training Time (per epoch)	164s	154s
	Inference Time	47.66s	57.69s
CICIDS2018	Accuracy	98.63%	98.38%
	Loss Rate	0.0050	0.0076
	∼Training Time (per epoch)	450s	440s
	Inference Time	105s	143.14s

TABLE 8. Performance comparison LSTM-AE with MAE error function.

With 5 epochs training Loss Function = MAE		Learning Rate	
Dataset	**Performance**	**0.0001**	**0.001**
CICIDS2017	Accuracy	93.87%	96.20%
	Loss	0.0042	0.0058
	∼Training Time (per epoch)	175s	170s
	Inference Time	47.74s	52.01s
CICIDS2018	Accuracy	98.76%	98.18%
	Loss Rate	0.0046	0.0056
	∼Training Time (per epoch)	470s	462s
	Inference Time	118.21s	133.94s

are data points that are significantly different from the majority of the data. In other words, MAE gives equal weight to all errors, while MSE heavily penalizes large errors because of the squaring term. This means that if there are outliers in the dataset, the model trained with MAE is less likely to be affected by them and more likely to generalize well to new data.

However, one disadvantage of using MAE is that it does not indicate the direction of the error. In this study, we balanced the high-bias class by pre-processing the dataset. In such cases, MAE may be more appropriate, which is why we selected MAE for our final evaluation.

Using a smaller dataset can reduce the computation required to train the LSTM-AE, and a lower learning rate can prevent overfitting and improve training stability. Therefore, we applied a learning rate of 0.001 for the optimization of our proposed model. A simpler AE architecture can also decrease the number of trainable parameters and make the model easier to train. Thus, our proposed model is both simple and effective for IDS.

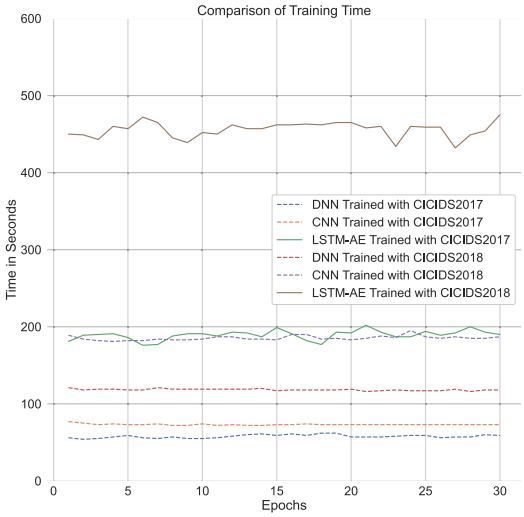


FIGURE 8. Time taken for training models.

VI. CONCLUSION AND FUTURE WORK

We propose a novel two-stage intrusion detection system in this article that utilizes a highly efficient framework and is capable of analyzing network activity. The system employs a distributed deep-learning model for real-time data processing and analysis, which includes the DNN model and CNN model that were selected for a thorough comparison with the proposed LSTM-AE model. We trained and tested the model using two datasets: CICIDS2017 and CSE-CICIDS2018. To the best of our knowledge, our system is capable of detecting malicious activity in a distributed manner and utilizes the proposed hybrid model to more precisely identify attacks. The proposed model has applications in various DL fields, such as agriculture, medicine, and language translation, and has shown a highly improved loss rate during training due to data cleaning. However, the proposed model also has limitations in terms of training time duration and complexity when training with a large dataset such as CICDDoS2019[5] as it still takes longer than typical DNN and CNN models, as shown in Table 5.

To evaluate the performance of the proposed hybrid model, we compared two error functions, MSE and MAE, with a training period of 5 epochs. The results showed that the MAE error function outperformed MSE in terms of accuracy performance. Therefore, we utilized the MAE error function for the proposed hybrid model with a longer training period of 30 epochs. It is worth noting that while MSE is more sensitive to large errors, it may not always be the best choice for all models or datasets. In some cases, such as with the intrusion detection system proposed in this study, MAE is a more appropriate choice due to its robustness to outliers and ability to provide a better estimate of the average error. The longer training period of 30 epochs was selected to ensure the model had sufficient time to learn and accurately detect

[5]CICDDoS2019 https://www.unb.ca/cic/datasets/ddos-2019.html

potential intrusions. The results of the experiments confirmed that the proposed hybrid model with MAE as the error function and 30 epochs of training achieved outstanding accuracy in detecting intrusion attempts in network traffic data, outperforming other state-of-the-art models for intrusion detection.

The proposed hybrid model achieved remarkable multi-class detection accuracy of 99.99% on the CICIDS2017 dataset, compared to 99.10% on the CSE-CICIDS2018 dataset when trained for up to 30 epochs. These experimental results outperformed those of other state-of-the-art intrusion detection models in terms of accuracy performance metrics.

The LSTM-AE model for IDS has a promising future with potential advancements that can further improve its performance. For instance, researchers can experiment with alternative architectures, such as stacked or bidirectional LSTM-AE models, to evaluate whether they can achieve better outcomes. Additionally, researchers can explore the application of attention mechanisms, which can help the model focus on essential features and ignore irrelevant ones, thereby boosting performance.

Another potential direction for the LSTM-AE model is to use transfer learning techniques. This technique involves pre-training the model on a large dataset and fine-tuning it on a smaller dataset, which could lead to improved accuracy and reduced training time.

Furthermore, researchers can consider employing ensemble methods, which involve combining multiple LSTM-AE models to enhance their overall performance. This approach can help to mitigate the impact of overfitting and improve the model's ability to generalize to new data.

Finally, as new types of attacks emerge in the field of network security, the LSTM-AE model may require adaptation to remain effective. Researchers can collect new datasets and retrain the model on them to ensure that it can continue to detect the latest threats. In summary, the LSTM-AE model has demonstrated great potential in intrusion detection and offers many opportunities for future development and improvement.

ABBREVIATIONS

The following abbreviations are used in this manuscript:

AE	AutoEncoder.
AI	Artificial Intelligence.
AIDS	Anomaly Intrusion Detection System.
BAU-ROA	Bypass-Linked Attacker Update-based ROA.
CAE	Computer-Aided Engineering.
CNN	Convolutional Neural Network.
CIC	Canadian Institute for Cybersecurity.
CPU	Control Processing Unit.
CSE	Communications Security Establishment
CSV	Comma-Separated Values.
DBN	Deep Belief Network.
DCNN	Deep Convolutional Neural Network.
DL	Deep Learning.
DNN	Deep Neural Network.
DoS	Denial of Service.
DDoS	Distributed Denial of Service.
FCL	Fully-Connected Layer.
FNN	Feedforward Neural Network.
GAN	Generative Adversarial Network.
GBT	Gradient Boosted Tree.
HMLSTM	Hierarchical Multi-scale LSTM.
HTTP	HyperText Transfer Protocol.
IDS	Intrusion Detection System.
KNN	k Nearest Neighbour.
LSTM	Long-Short Term Memory.
MCTS	Monte Carlo Tree Search.
ML	Machine Learning.
MLP	Multi Layer Perceptron.
NB	Naive Bayes.
NDAE	Non-symmetric Deep AutoEncoder.
OCNN	Optimised Convolutional Neural Network.
OCSVM	One Class Support Vector Machine.
PCA	Principal Component Analysis.
PL	Pooling Layer.
RAM	Random Access Memory.
RBM	Restricted Boltzmann Machine.
ReLU	Rectified Linear Unit.
RF	Random Forest.
RNN	Recurrent Neural Network.
ROA	Rider Optimization Algorithm.
SAE	Sparse AutoEncoder.
SMOTE	Synthetic Minority Oversampling Technique.
SQL	Structure Query Language.
SVM	Support Vector Machine.
TAP	Terminal Access Point.
VARMAN	adVanced multi-plAne secuRity fraMework. for softwAre defined Networks.
XSS	Cross Site Scripting.

REFERENCES

[1] D. Chou and M. Jiang, "A survey on data-driven network intrusion detection," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–36, Oct. 2021, doi: 10.1145/3472753.

[2] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Netw.*, vol. 8, no. 3, pp. 26–41, May 1994, doi: 10.1109/65.283931.

[3] G. D. C. Bertoli, L. A. Pereira Jr., O. Saotome, A. L. D. Santos, F. A. N. Verri, C. A. C. Marcondes, S. Barbieri, M. S. Rodrigues, and J. M. P. D. Oliveira, "An end-to-end framework for machine learning-based network intrusion detection system," *IEEE Access*, vol. 9, pp. 106790–106805, 2021, doi: 10.1109/ACCESS.2021.3101188.

[4] G. Apruzzese, L. Pajola, and M. Conti, "The cross-evaluation of machine learning-based network intrusion detection systems," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 5152–5169, Dec. 2022, doi: 10.1109/TNSM.2022.3157344.

[5] V. Hnamte and J. Hussain, "DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system," *Telematics Informat. Rep.*, vol. 10, Jun. 2023, Art. no. 100053, doi: 10.1016/j.teler.2023.100053.

[6] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 686–728, 1st Quart., 2019, doi: 10.1109/COMST.2018.2847722.

[7] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *Knowl.-Based Syst.*, vol. 136, pp. 130–139, Nov. 2017, doi: 10.1016/j.knosys.2017.09.014.

[8] W. Meng, W. Li, and L.-F. Kwok, "Design of intelligent KNN-based alarm filter using knowledge-based alert verification in intrusion detection," *Secur. Commun. Netw.*, vol. 8, no. 18, pp. 3883–3895, Dec. 2015, doi: 10.1002/sec.1307.

[9] S. Mukherjee and N. Sharma, "Intrusion detection using naive Bayes classifier with feature reduction," *Proc. Technol.*, vol. 4, pp. 119–128, Feb. 2012, doi: 10.1016/j.protcy.2012.05.017.

[10] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Proc. Comput. Sci.*, vol. 89, pp. 213–217, May 2016, doi: 10.1016/j.procs.2016.06.047.

[11] J. Hussain and V. Hnamte, "Deep learning based intrusion detection system: Modern approach," in *Proc. 2nd Global Conf. Advancement Technol. (GCAT)*, Oct. 2021, pp. 1–6, doi: 10.1109/GCAT52182.2021.9587719.

[12] J. Jiang, M. Chen, and J. A. Fan, "Deep neural networks for the evaluation and design of photonic devices," *Nature Rev. Mater.*, vol. 6, pp. 679–700, Dec. 2020, doi: 10.1038/s41578-020-00260-1.

[13] X. Kan, Y. Fan, Z. Fang, L. Cao, N. N. Xiong, D. Yang, and X. Li, "A novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network," *Inf. Sci.*, vol. 568, pp. 147–162, Aug. 2021, doi: 10.1016/j.ins.2021.03.060.

[14] M. Sheikhan, T. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size RNN based on feature grouping," *Neural Comput. Appl.*, vol. 21, no. 6, pp. 1185–1190, Sep. 2012, doi: 10.1007/s00521-010-0487-0.

[15] S. A. Althubiti, E. M. Jones, and K. Roy, "LSTM for anomaly-based network intrusion detection," in *Proc. 28th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Nov. 2018, pp. 1–3, doi: 10.1109/ATNAC.2018.8615300.

[16] Y. Liu, S. Liu, and X. Zhao, "Intrusion detection algorithm based on convolutional neural network," *DEStech Trans. Eng. Technol. Res.*, vol. 37, no. 12, pp. 1271–1275, 2018, doi: 10.12783/dtetr/iceta2017/19916.

[17] S. Alzahrani and L. Hong, "Detection of distributed denial of service (DDoS) attacks using artificial intelligence on cloud," in *Proc. IEEE World Congr. Services (SERVICES)*, Jul. 2018, pp. 35–36, doi: 10.1109/SERVICES.2018.00031.

[18] A. Kim, M. Park, and D. H. Lee, "AI-IDS: Application of deep learning to real-time web intrusion detection," *IEEE Access*, vol. 8, pp. 70245–70261, 2020, doi: 10.1109/ACCESS.2020.2986882.

[19] X. Zhang, Y. Zhou, S. Pei, J. Zhuge, and J. Chen, "Adversarial examples detection for XSS attacks based on generative adversarial networks," *IEEE Access*, vol. 8, pp. 10989–10996, 2020, doi: 10.1109/ACCESS.2020.2965184.

[20] T. A. S. Srinivas and S. S. Manivannan, "Prevention of hello flood attack in IoT using combination of deep learning with improved rider optimization algorithm," *Comput. Commun.*, vol. 163, pp. 162–175, Nov. 2020, doi: 10.1016/j.comcom.2020.03.031.

[21] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN," *Future Gener. Comput. Syst.*, vol. 111, pp. 763–779, Oct. 2020, doi: 10.1016/j.future.2019.10.015.

[22] R. Priyadarshini and R. K. Barik, "A deep learning based intelligent framework to mitigate DDoS attack in fog environment," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 3, pp. 825–831, Mar. 2022, doi: 10.1016/j.jksuci.2019.04.010.

[23] P. Krishnan, S. Duttagupta, and K. Achuthan, "VARMAN: Multi-plane security framework for software defined networks," *Comput. Commun.*, vol. 148, pp. 215–239, Dec. 2019, doi: 10.1016/j.comcom.2019.09.014.

[24] P. R. Kanna and P. Santhi, "Unified deep learning approach for efficient intrusion detection system using integrated spatial–temporal features," *Knowl.-Based Syst.*, vol. 226, Aug. 2021, Art. no. 107132, doi: 10.1016/j.knosys.2021.107132.

[25] J. Hussain and V. Hnamte, "A novel deep learning based intrusion detection system: Software defined network," in *Proc. Int. Conf. Innov. Intell. Informat., Comput., Technol. (3ICT)*, Sep. 2021, pp. 506–511, doi: 10.1109/3ICT53449.2021.9581404.

[26] S. N. Mighan and M. Kahani, "A novel scalable intrusion detection system based on deep learning," *Int. J. Inf. Secur.*, vol. 20, no. 3, pp. 387–403, Jun. 2021, doi: 10.1007/s10207-020-00508-5.

[27] G. Lu and X. Tian, "An efficient communication intrusion detection scheme in AMI combining feature dimensionality reduction and improved LSTM," *Secur. Commun. Netw.*, vol. 2021, pp. 1–21, Apr. 2021, doi: 10.1155/2021/6631075.

[28] A. Binbusayyis and T. Vaiyapuri, "Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class SVM," *Appl. Intell.*, vol. 51, pp. 7094–7108, Feb. 2021, doi: 10.1007/s10489-021-02205-9.

[29] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64375–64387, 2022, doi: 10.1109/ACCESS.2022.3182333.

[30] N. Wang, Y. Chen, Y. Xiao, Y. Hu, W. Lou, and Y. Thomas Hou, "MANDA: On adversarial example detection for network intrusion detection system," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 2, pp. 1139–1153, Apr. 2023, doi: 10.1109/TDSC.2022.3148990.

[31] M. B. Umair, Z. Iqbal, M. A. Faraz, M. A. Khan, Y.-D. Zhang, N. Razmjooy, and S. Kadry, "A network intrusion detection system using hybrid multilayer deep learning model," *Big Data*, 2022, doi: 10.1089/big.2021.0268.

[32] V. Ravi, R. Chaganti, and M. Alazab, "Recurrent deep learning-based feature fusion ensemble meta-classifier approach for intelligent network intrusion detection system," *Comput. Electr. Eng.*, vol. 102, Sep. 2022, Art. no. 108156, doi: 10.1016/j.compeleceng.2022.108156.

[33] G. Bae and I. Joe, "UAV anomaly detection with distributed artificial intelligence based on LSTM-AE and AE," in *Advanced Multimedia and Ubiquitous Engineering*, J. J. Park, L. T. Yang, Y.-S. Jeong, and F. Hao, Eds. Singapore: Springer, 2020, pp. 305–310, doi: 10.1007/978-981-32-9244-4_43.

[34] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, "A survey on anomaly detection for technical systems using LSTM networks," *Comput. Ind.*, vol. 131, Oct. 2021, Art. no. 103498, doi: 10.1016/j.compind.2021.103498.

[35] A. S. Musleh, G. Chen, Z. Yang Dong, C. Wang, and S. Chen, "Attack detection in automatic generation control systems using LSTM-based stacked autoencoders," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 153–165, Jan. 2023, doi: 10.1109/TII.2022.3178418.

[36] E. Mushtaq, A. Zameer, M. Umer, and A. A. Abbasi, "A two-stage intrusion detection system with auto-encoder and LSTMs," *Appl. Soft Comput.*, vol. 121, May 2022, Art. no. 108768, doi: 10.1016/j.asoc.2022.108768.

[37] M. Mahmoud, M. Kasem, A. Abdallah, and H. S. Kang, "AE-LSTM: Autoencoder with LSTM-based intrusion detection in IoT," in *Proc. Int. Telecommun. Conf. (ITC-Egypt)*, Jul. 2022, pp. 1–6, doi: 10.1109/ITC-Egypt55520.2022.9855688.

[38] H. C. Altunay and Z. Albayrak, "A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks," *Eng. Sci. Technol., Int. J.*, vol. 38, Feb. 2023, Art. no. 101322, doi: 10.1016/j.jestch.2022.101322.

[39] A. S. A. Issa and Z. Albayrak, "DDoS attack intrusion detection system based on hybridization of CNN and LSTM," *Acta Polytechnica Hungarica*, vol. 20, no. 2, pp. 1–19, 2023, doi: 10.12700/APH.20.2.2023.2.6.

[40] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[41] V. Hnamte and J. Hussain, "An extensive survey on intrusion detection systems: Datasets and challenges for modern scenario," in *Proc. 3rd Int. Conf. Electr., Control Instrum. Eng. (ICECIE)*, Nov. 2021, pp. 1–10, doi: 10.1109/ICECIE52348.2021.9664737.

[42] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 4487–4492, doi: 10.1109/WCNC.2013.6555301.

[43] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Secur.*, vol. 86, pp. 147–167, Oct. 2019, doi: 10.1016/j.cose.2019.06.005.

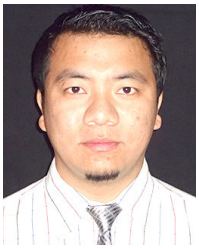
[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[45] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3.24, pp. 479–482, 2018, doi: 10.14419/ijet.v7i3.24.22797.

[46] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami, "Multi-stage optimized machine learning framework for network intrusion detection," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1803–1816, Jun. 2021, doi: 10.1109/TNSM.2020.3014929.

[47] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[48] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.
[49] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, vol. 1, Jan. 2018, pp. 108–116, doi: 10.5220/0006639801080116.
[50] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation," *J. Mach. Learn. Technol.*, vol. 2, pp. 37–63, Dec. 2011.
[51] K. K. Nguyen, D. T. Hoang, D. Niyato, P. Wang, D. Nguyen, and E. Dutkiewicz, "Cyberattack detection in mobile cloud computing: A deep learning approach," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6, doi: 10.1109/WCNC.2018.8376973.
[52] W. Wang, "HAST-IDS: Learning hierarchical spatial–temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018, doi: 10.1109/ACCESS.2017.2780250.
[53] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving AdaBoost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset," *J. Phys., Conf. Ser.*, vol. 1192, Mar. 2019, Art. no. 012018, doi: 10.1088/1742-6596/1192/1/012018.
[54] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, 2019, doi: 10.3390/electronics8030322.
[55] N. Chouhan, A. Khan, and H.-U.-R. Khan, "Network anomaly detection using channel boosted and residual learning based deep convolutional neural network," *Appl. Soft Comput.*, vol. 83, Oct. 2019, Art. no. 105612, doi: 10.1016/j.asoc.2019.105612.
[56] S. Kaur and M. Singh, "Hybrid intrusion detection and signature generation using deep recurrent neural networks," *Neural Comput. Appl.*, vol. 32, no. 12, pp. 7859–7877, Jun. 2020, doi: 10.1007/s00521-019-04187-9.
[57] S. Huang and K. Lei, "IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks," *Ad Hoc Netw.*, vol. 105, Aug. 2020, Art. no. 102177, doi: 10.1016/j.adhoc.2020.102177.
[58] Z. Pelletier and M. Abualkibash, "Evaluating the CIC IDS-2017 dataset using machine learning methods and creating multiple predictive models in the statistical computing language R," *Science*, vol. 5, no. 2, pp. 187–191, 2020. [Online]. Available: http://irjaes.com/wp-content/uploads/2020/10/IRJAES-V5N2P184Y20.pdf



VANLALRUATA HNAMTE (Graduate Student Member, IEEE) received the B.C.A. degree from Makhanlal Chaturvedi National University for Journalism and Communication, Bhopal, India, in 2009, and the M.C.A. degree from Annamalai University, India, in 2011. He is currently pursuing the Ph.D. degree with the Department of Mathematics and Computer Science, Mizoram University, under the supervision of Prof. Jamal Hussain. In 2012, he qualified for the National Eligibility Test and was awarded a Lectureship by the University Grants Commission. His research interests include artificial intelligence, machine learning, deep learning, network security, cyber security, and machine-automated language translation.

HONG NHUNG-NGUYEN (Graduate Student Member, IEEE) received the B.S. degree in information technology and the master's degree in software engineering from Ha Noi National University, Ha Noi, Vietnam, in 2015 and 2018, respectively. She is currently pursuing the Ph.D. degree with the Information Technology Convergence Laboratory, Department of Electronic Engineering, Myongji University (MJU), South Korea, where she is advised by Prof. Yong Hwa-Kim. Since 2016, she has been a Lecturer with the Faculty of Information Technology, Viet Tri University of Industry, Vietnam. Her research interests include machine learning and software engineering.

JAMAL HUSSAIN received the B.Sc. degree in mathematics from Dibrugarh University, Assam, India, in 1993, and the M.Sc. and Ph.D. degrees from Tezpur University, Assam, in 1996 and 2000, respectively. He has been a Professor with the Department of Mathematics and Computer Science, Mizoram University, since 2006. He has organized more than 30 international conferences and has successfully guided ten Ph.D. scholars. In addition, he has completed a project titled "Applicability of Artificial Neural Network for Intrusion Detection" funded by the Department of Information Technology, Ministry of Communication and Information Technology, Government of India. His research interests include mathematical modeling, biological computer simulation, ecological and environmental systems, artificial intelligence, deep learning, network security, and intrusion detection.

YONG HWA-KIM (Member, IEEE) received the B.S. degree in electrical engineering and the Ph.D. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2001 and 2007, respectively. From 2007 to 2011, he was a Senior Researcher with the Korea Electrotechnology Research Institute (KERI), Gyeonggi, South Korea. From 2011 to 2013, he was an Assistant Professor with the Division of Maritime Electronic and Communication Engineering, Mokpo National Maritime University, South Korea. From 2013 to 2021, he was a Professor with the Department of Electronic Engineering, Myongji University, South Korea. In April 2021, he joined the Korea National University of Transportation, as a Faculty Member. His research interests include communication systems, fault diagnoses, and digital signal processing. Currently, he is interested in artificial intelligence for communication, radar systems, and smart grids.