Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

An Improved Network Intrusion Detection Method Based On CNN-LSTM-SA

Bian Hui[1, 2], Kang Leng Chiew[2*]

[1] Qinyuan (Jiangsu) Technology Co., Ltd., China
[2] Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Malaysia

ARTICLE INFO

Article history:
Received 12 October 2023
Received in revised form 8 December 2023
Accepted 19 March 2024
Available online 25 April 2024

Keywords:
Network intrusion detection; Machine learning; Deep learning; Long short-term memory

ABSTRACT

Network intrusion detection is an essential component of contemporary cybersecurity strategies, and the development of efficient techniques to accurately identify malicious activities has become a priority. This study investigates the performance of various conventional machine learning algorithms, including decision trees, naive Bayes, naive Bayes trees, random forest, random trees, MLP, and SVM, in detecting network intrusions using binary and multi-classification approaches. Furthermore, the study proposes a deep learning method, CNN-LSTM-SA, which consistently outperforms conventional machine learning techniques in terms of precision, recall, F1 score, and overall accuracy for network intrusion detection. Specifically, the proposed method combines CNN and LSTM with SA in machine learning theory to extract more optimized, strongly correlated features. The proposed method is evaluated using the benchmark NSL-KDD database. The results indicate that the CNN-LSTM-SA method holds great potential in enhancing the efficacy of network intrusion detection systems.

1. Introduction

The Internet has become the main infrastructure for people's daily lives and work, and its importance in politics, culture, economy, military and other fields has been continuously demonstrated. It has become a powerful driving force for promoting economic and social development in the 21st century [1-3]. With the popularization of the Internet, various network services and applications have emerged and penetrated into all aspects of people's work and life. Along with the rapid development of the Internet and Internet-based applications, the network environment has become increasingly complex, resulting in many issues related to network security. Intrusion detection, as a major component of network security measures, can actively defend against network attacks and respond before the network is harmed, playing a very important role. In today's rapidly developing information technology era, it is an urgent task to effectively combat attacks and ensure the security of Internet-based information systems [4]. Due to the complexity and diversity of attackers' behaviors, it is very difficult to model both normal and attack behaviors by identifying

[*] *Corresponding author.*
E-mail address: klchiew@unimas.my

https://doi.org/10.37934/araset.44.1.225238

correlations and summarizing patterns from massive network access behaviors [5]. Therefore, research on intrusion detection systems has always been an important content of network security research. Studying intrusion detection methods and improving the detection accuracy of intrusion detection systems have significant scientific and social significance [6].

As machine learning technology has become increasingly mature, it has emerged as a potential solution [7]. The vast amount of heterogeneous big data originating from various sources presents a challenge for conventional data analytics and shallow machine learning (ML) techniques, rendering them inadequate for addressing security concerns [8]. It is worth noting that conventional ML methods face limitations in terms of latency, computational complexity, and the ability to learn complex, time-varying, and nonlinear relationships within large datasets [9-11].

To address these issues, this paper proposed a combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs) with self-attention mechanism (CNN-LSTM-SA) to process network intrusion behavior. Section 2 will discuss about the related work. Section 3 presents material and method used in this paper, including the description of dataset, conventional and proposed machine learning method. In Section 4, all the experiments' outcomes are examined, and the concluding remarks with some recommendations for further work are presented in Section 5.

2. Related Work

The concept of intrusion detection originated from a technical report entitled "Computer security threat monitoring and surveillance" in 1980 by Anderson [12], which detailed the concept of intrusion detection for the first time. A groundbreaking work conducted by Xiao *et al.,* [13] suggested a basic model for intrusion detection, which had officially launched the research in the field of intrusion detection. Since then, many researchers have conducted intrusion detection studies for different computer systems and network environments, achieving significant results [14]. Computer and network security protection systems now typically include intrusion detection systems as a crucial component of their overall architecture. Many pertinent methods have been applied to the network intrusion detection since the introduction of data mining and machine learning. In the topic of intrusion detection, research on intrusion detection algorithms that are based on data mining and machine learning has continued to be a significant study issue up until the present day.

Vitrià *et al.,* [15] proposed a hybrid method combining genetic algorithms and decision trees to detect network intrusions. They used a genetic algorithm to optimize decision tree parameters and effectively classify normal and attack activities. The method achieved high detection rates and low false alarm rates, but it may suffer from increased computational complexity due to the genetic algorithm component.

Kim et al., [16] proposed a deep learning-based Network Intrusion Detection System (NIDS) using CNNs. The authors showed that the CNNs model could automatically learn and extract relevant features from raw network data, resulting in high detection accuracy. However, the model's primary disadvantage is that it requires significant computation resources and training time to achieve optimal results.

Garcia *et al.,* [17] and Catania and Garino [18] both presented ensemble-based approaches to network intrusion detection. Ensemble methods, such as bagging and boosting, were employed to combine multiple classifiers, improving overall detection accuracy and reducing false positives. The primary disadvantage of these approaches is the increased complexity and computational requirements associated with using multiple classifiers.

Mukkamala *et al.,* [19] applied SVM to network intrusion detection, demonstrating its effectiveness in classifying different types of attacks. SVMs are particularly suitable for intrusion detection due to their ability to handle high-dimensional datasets and their generalization capabilities. However, SVMs can be computationally expensive, especially for large datasets, and may not be well-suited for real-time detection.

Feature selection and dimensionality reduction have been prominent research areas in NIDS. Alazab *et al.,* [20] and Wang *et al.,* [21] applied techniques like mutual information and principal component analysis (PCA) to reduce the feature space's size and enhance the detection performance of classifiers. These methods helped improve the efficiency and effectiveness of NIDS, but their success depends on the quality of selected features.

Deep learning techniques, such as CNNs and recurrent neural networks (RNNs), have become increasingly popular in NIDS. Vinayakumar *et al.,* [22] and Brown *et al.,* [23] demonstrated the effectiveness of deep learning in learning complex patterns and extracting relevant features from raw network data. However, these approaches require significant computational resources and training time, posing challenges for real-time applications.

Ensemble techniques have gained attention for improving the performance of intrusion detection systems. Yin *et al.,* [24] and Ahmad *et al.,* [25] utilized methods like bagging, boosting, and stacking to combine multiple classifiers, resulting in better detection accuracy and reduced false positives. Despite their effectiveness, these approaches may suffer from increased complexity and computational requirements.

Benaddi *et al.,* [26] applied reinforcement learning algorithms to train NIDS that can adapt to evolving network environments and dynamically update their detection strategies. This adaptability enables NIDS to counter new and previously unseen attacks. However, reinforcement learning-based NIDS can suffer from convergence issues and may require longer training times compared to other machine learning techniques.

In summary, deep learning-based network anomaly detection technology has become a major research direction and hotspot in the field of intrusion detection and botnet detection, but there are still many unresolved issues in the existing research: First, deep learning is mostly used in the pre-training stage, and related research mainly uses deep learning algorithms as generative models and traditional supervised machine learning methods in combination, without a comprehensive separate research on the detection and classification capabilities of deep learning. Secondly, most of the studies focus on the binary classification task, this aspect has less of an impact on the efficacy of deep learning when it is used to tackle complex multi-classification tasks. Third, both the theory and the technology behind deep learning are advancing at a rapid rate. Despite the advent of hybrid neural networks and attention mechanisms in recent years, it is still important to keep up with the most recent findings in the field of deep learning research and to investigate how deep learning can be applied in network anomaly detection technology. Thus, this paper proposed a combination of CNNs and LSTM-RNNs with self-attention mechanism (CNN-LSTM-SA) to process network intrusion behavior to address these issues.

3. Material and Method

3.1 NSL-KDD Database Description and Preprocessing

Since the KDD CUP 99 dataset was used as a knowledge discovery competition dataset, it has been one of the most commonly used datasets for evaluating network anomaly detection methods in the field of intrusion detection. The dataset is based on network traffic captured by the U.S. Department of Defense with about seven weeks of network traffic captured as training data and it has approximately five million connection records. Another two weeks of network data were captured as testing data, with about two million connection records. There are five categories in the KDD CUP 99 dataset with one normal and four abnormal categories (i.e., Dos, Probe, R2L, U2R). For the four abnormal categories, there are a total of 39 attack types, of which 22 are included in the training set, and the remaining 17 attack types are reserved for the testing set. This distribution is to ensure that the dataset can be fairly and effectively used to evaluate the model's ability to detect unknown attacks. The five categories and their attack types of KDD CUP 99 dataset are shown in Table 1. In the KDD CUP 99 dataset, for each row of data, there are 41 fixed features, and a category.

Table 1
Categories and their attack types of the KDD CUP 99 dataset

Normal	DoS	Probe	R2L	U2R
	apache2	ipsweep	Spy	bufferoverflow loadmodule
	back	mscan	warezclient	perl
	land	nmap	ftp_ write guesspasswd httptunnel	ps
	mailbomb	portsweep	imap	rootkit
	Neptune	saint	multihop	snmpguess sqlattack
	pod	satan	named	worm
	processtable		phf	xterm
	smurf		sendmail snmpgetattack warezmaster	
	teardrop		xlock	
	udpstorm		xsnoop	

The major concern in the KDD Cup 99 dataset is the presence of a large number of redundant records, which causes the trained algorithm model to be biased towards categories with more labeled samples, thus greatly affecting the actual detection performance of the model. In addition, the large number of duplicate records in the testing set also affect the evaluation results. For example, the smurf and neptune attacks from DoS category, which have accounted for more than 71% of the testing set can be easily detected by any intrusion detection system. Clearly, this dataset is not appropriate to be used for a proper evaluation of model performance.

A better dataset will be the NSL-KDD dataset and it will be used in this paper. This dataset has solved the problem of redundant records in the KDD Cup 99 dataset and optimized the record configuration of each category [27]. Compare to KDD CUP 99, the 43rd column is added in NSL-KDD dataset to measure the difficulty of predicting corresponding categories. The NSL-KDD dataset includes two training sets, KDDTrain+ and KDDTrain+20%, and two testing sets, KDDTest+ and KDDTest-21. Table 2 summarizes the five categories of label data in the training and testing sets. KDDTrain+20% is a 20% subset of the KDDTrain+ set, and KDDTest-21 is a subset obtained by removing records marked as 0-21 in the 43rd column of the KDDTest+ set. The records in this KDDTest-21 are relatively difficult to detect.

Table 2
Samples sizes and proportions of categories in the NSL-KDD dataset

Dataset	Total	Normal	DoS	Probe	U2R	R2L
KDDTrain+20%	25192	13449 (53.00%)	9234 (37.00%)	2289 (9.16%)	11 (0.04%)	209 (0.80%)
KDDTrain+	125973	67343 (53.00%)	45927 (37.00%)	11656 (9.11%)	52 (0.04%)	995 (0.85%)
KDDTest+	22544	9711 (43.00%)	7458 (33.00%)	2421 (11.00%)	200 (0.90%)	2754 (12.10%)
KDDTest-21	11 850	2152 (18.00%)	4342 (37.00%)	2402 (20.00%)	200 (2.00%)	2754 (23.00%)

In order to use the NSL-KDD dataset in the proposed method, data preprocessing is necessary. As mentioned before, NSL-KDD has 41 fixed features, which include 34 continuous attributes, 4 binary attributes and 3 symbolic attributes (including protocols, flags, service). Based on the 3 symbolic attributes, a set of 84 features are further derived. Namely, 3 features of one-hot encoding from protocols, 11 features from flags and 70 features from service. Therefore, the features set from NSL-KDD will be 122 features (i.e., 34+4+3+11+70=122). The "num_outbound_crnds", which is one of the 122 features is observed to have the value of zero in all the samples. This feature will be removed and resulted in a final features set of 121.

Since convolutional neural networks can only handle two-dimensional data, the final features set are transformed into an 11x11 matrix.

The data preprocessing steps are as follows:

(1) Convert symbolic features to numerical features using attribute mapping based on one-hot encoding.
(2) Normalize the numerical data obtained in step 1 with the normalization formula as shown in Eq. (1) to get the range of [0,1].

$$y_i = \frac{x_i - m_{min}}{m_{max} - m_{min}} \tag{1}$$

where x_i is the i-th attribute, x_{min} and x_{max} is the smallest and largest value of the attribute record, respectively.

(3) Different binary numbers are used to represent different intrusion behaviors: 000 represents DoS, 001 represents Normal, 010 represents Probe, 011 represents R2L, and 100 represents U2R. This completes the labeling of intrusion types. After the preprocessing of intrusion records, the original 41 features discussed were transformed into 121 features. To meet the input data format requirements of convolutional neural networks, the one-dimensional feature data is transformed into a two-dimensional feature matrix with a size of 11 × 11, as shown in Figure 1.

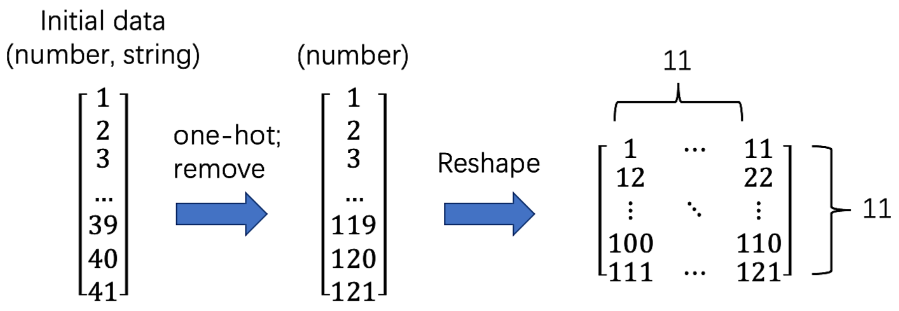

Fig. 1. The processing of attribute mapping for CNN-LSTM-SA method

3.2 Proposed CNN-LSTM-SA

Based on the previous study conducted by Krizhevsky *et al.,* [28], the CNNs are a form of deep learning technique that are widely utilized for image processing and computer vision tasks. CNNs are built with many convolutional layers, pooling layers, and fully connected layers. These layers are responsible for automatically extracting features from raw pixel data. Figure 2 illustrates a simple convolutional neural network structure. The convolutional layers extract local features of the image using convolutional filters, while the pooling layers reduce the dimensionality of the feature maps. The fully connected layers transform the feature maps into the final prediction. CNNs can handle high-dimensional data, and are robust to translations, rotations, and scaling of images, making them widely used in tasks such as image classification, object detection, and image segmentation.

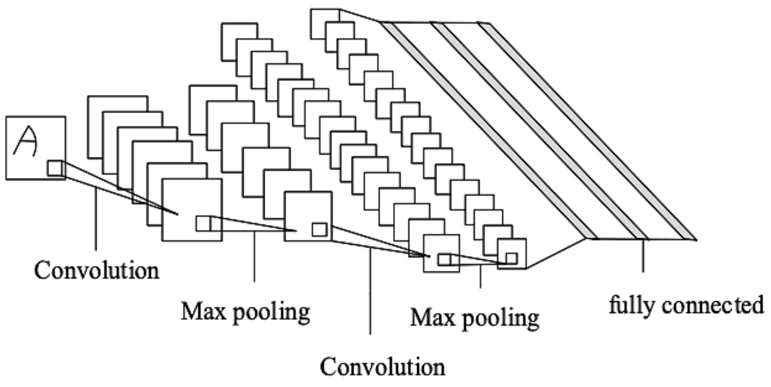

Fig. 2. Simple convolutional neural network structure

The LSTM by Hochreiter and Schmidhuber [29] is a typical RNNs model that is used mostly for processing sequential data such as speech recognition, natural language processing, and time series prediction tasks. The model of LSTM is as shown in Figure 3. The σ (sigma) parameter in Figure 3 refers to the sigmoid activation function, which is commonly used in the gating mechanisms of an LSTM network. A real-valued input is processed by the sigmoid function, which then returns a value in the range of 0 to 1. By gating the input, output, and forget actions, this function is utilized to control the flow of information that is transmitted throughout the network. This is a reference to the hyperbolic tangent activation function, which is also frequently utilized in an LSTM network. It is abbreviated as tanh. A real-valued input is processed by the tanh function, which then returns a value in the range of-1 to 1. This function is used to construct the candidate activation vector as well as to update the current state of the cell. The parameter x refers to the input to the LSTM network at a given time step. It is a vector of features that represent the input data. Whereas the symbol + refers to the element-wise addition operation, which is used in the LSTM network to combine the input with the previous hidden state. In an LSTM network, these terms are combined to perform the operations necessary for the network to process sequential data. Specifically, the network uses the sigmoid function to decide which information to keep or discard from the previous hidden state, the tanh function to update the cell state, and the element-wise addition to combine the input with the previous hidden state. Compared to conventional RNNs, LSTM addresses the problems of handling long sequences and vanishing/exploding gradients by introducing gating mechanisms. Every LSTM cell has three gates: the input gate, the forget gate, and the output gate. These gates are used to process information. The input gate is responsible for controlling the insertion of new information, the forget gate is responsible for controlling the retention of old information, and the output gate is responsible for controlling the output during the currently active time step. By jointly controlling these gates, LSTM can balance long-term and short-term memory and better handle sequential data.

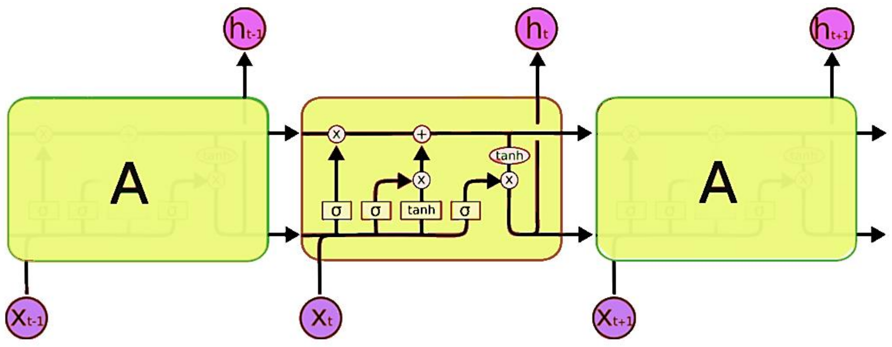

Fig. 3. Network structure of LSTM

Self-attention studied by Vaswani *et al.,* [30] is one of the mechanisms used in deep learning model and is largely utilized in natural language processing tasks like machine translation and text categorization. During processing, the model is able to concentrate its attention on various components of the input sequence, which also assists in the identification of long-range dependencies. Long-range dependencies refer to the relationships between elements in a sequence that are far apart. In natural language processing tasks, for instance, a word at the beginning of a sentence may influence the interpretation of a word appearing much later. Deep learning models with self-attention mechanisms are designed to capture these relationships, thereby significantly improving their performance on tasks such as machine translation and text categorization. The self-attention mechanism works by computing the similarity between all pairs of positions in the input sequence and using the resulting scores to weight the contributions of each position to the final output. It has been demonstrated that self-attention is an effective strategy for enhancing performance in a variety of natural language processing tasks.

To optimize the performance of network intrusion detection and improve the accuracy while reducing the false positive rate, this paper proposes the CNN-LSTM-SA method.

Firstly, three blocks of convolutional layer and max pooling are used to form the CNN (Figure 5 is referred). Input to the CNN is the 11x11 features matrix. The introduction of a batch normalization layer to the CNN network will enhance the training speed and effectiveness of CNN, and mitigate the influence of starting parameters on the training process.

Secondly, to address the impact of the temporal features of the preceding and following feature points on each attribute, an LSTM model consisting of memory modules is used for long-distance dependency feature extraction. The previous obtained feature map from CNN is input into LSTM network to mine the hidden relationship between features and temporal features.

Finally, the attention mechanism is used in order to determine the weight that should be given to each attribute feature, and a Softmax classifier is utilized in order to get the results of the classification.

The flow chart of the proposed method is shown in Figure 4. The structure of the proposed CNN-LSTM-SA is shown in Figure 5 (The number inside the parentheses represents the number of filters, and convolution kernel size. Note that this is an optimized structure and the optimization process will be discussed in Section 4. 1).

The number of convolution kernels is increased from 32 to 128, in deep learning, especially in CNNs, lower layers tend to learn low-level features such as edges and textures, while deeper layers capture high-level, more abstract features like shapes or specific objects. By increasing the number of filters, CNNs enable the model to learn a wider range of high-level features.

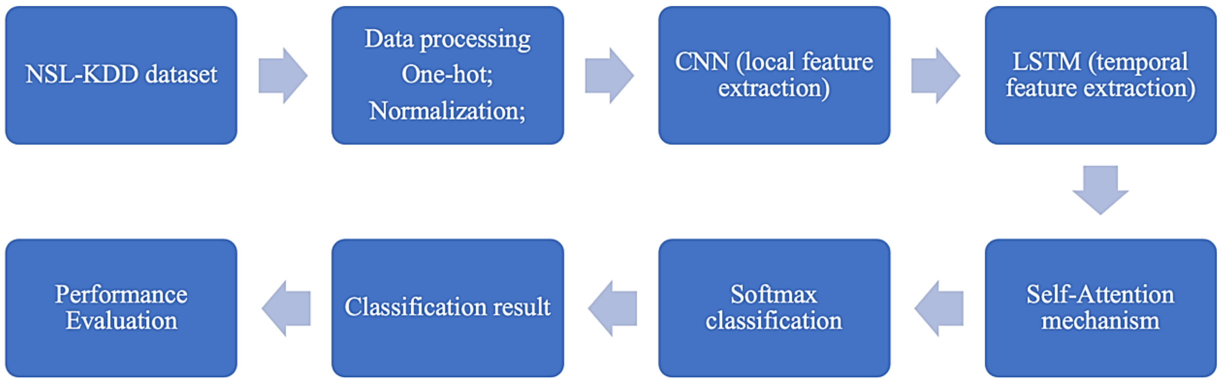

Fig. 4. Flow chart of CNN-LSTM-SA based network intrusion detection

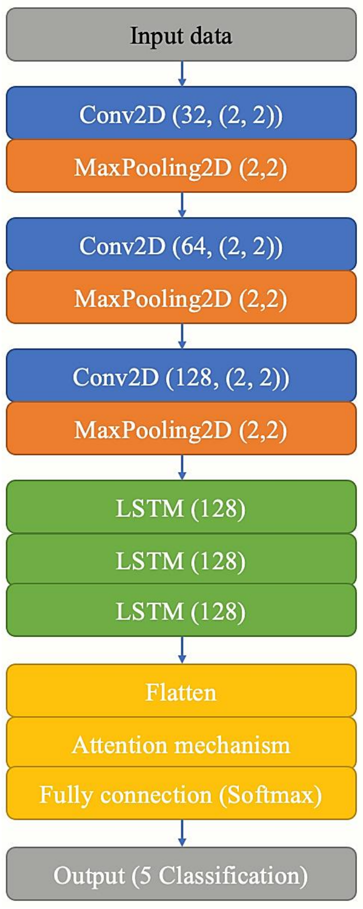

Fig. 5. The structure of the proposed CNN-LSTM-SA

4. Results and Discussions

4.1 Experimental Setup

The machine learning process is executed on a high-performance GPU GeForce RTX 2080 Ti and 16 cores Intel(R) Core (TM) i7-8000K CPU processor with 64 GB RAM. The training and testing sets are KDDTrain+ and KDDTest+, respectively. Python 3.8 and Pytorch 2.0 are used.

During the model training process, we found that the size of convolutional kernels, the number of hidden layers of LSTM and the number of neurons per layer have some influence on the error of the training set, and we tuned these parameters to get the model with the best results. The Adam optimizer is selected with the learning rate is set to 0.001 as the default value and the training epoch is set to 3000. The range of hyperparameters which will be used is shown in Table 3 and one of the combinations with the least training error will be selected as the optimum setting. After the training has completed, the model with the smallest training error within 3000 steps was selected as the best model, each training experiment took about 6 hours. Table 4 shows the list of combinations based on the hyperparameters range in Table 3 and their respective training errors.

Table 3
The range of hyperparameters

Item	Range
The number of convolutional kernels	(2, 2), (3, 3), (4, 4)
The number of hidden layers of LSTM	3, 4, 5
The number of neurons per layer	32, 64, 128

Table 4
Hyperparameters and their respective training error

Combination	Train error
(2, 2), 3, 32	0.1419
(2, 2), 3, 64	0.3653
(2, 2), 3, 128	0.0622
(2, 2), 4, 32	0.4964
(2, 2), 4, 64	0.3557
(2, 2), 5, 128	0.1277
(2, 2), 5, 32	0.1088
(2, 2), 5, 64	0.5200
(2, 2), 5, 128	0.0785
(3, 3), 3, 32	0.1795
(3, 3), 3, 64	0.2585
(3, 3), 3, 128	0.3292
(3, 3), 4, 32	0.4170
(3, 3), 4, 64	0.4328
(3, 3), 4, 128	0.5895
(3, 3), 5, 32	0.1941
(3, 3), 5, 64	0.3339
(3, 3), 5, 128	0.2236
(4, 4), 3, 32	0.2390
(4, 4), 3, 64	0.4349
(4, 4), 3, 128	0.3729
(4, 4), 4, 32	0.1980
(4, 4), 4, 64	0.2173
(4, 4), 4, 128	0.4381
(4, 4), 5, 32	0.1787
(4, 4), 5, 64	0.0916
(4, 4), 5, 128	0.3590

From Table 4, the combination with convolutional kernels of (3, 3), hidden layers of LSTM of 3 and number of neurons per layer of 128 has the best performance. Thus, the proposed CNN-LSTM-SA will be using this combination of hyperparameters for all the classification tasks.

In order to evaluate the performance of the proposed CNN-LSTM-SA, this paper will use seven conventional machine learning algorithms classifiers, namely decision trees studied by Breiman *et al.,* [31], naive Bayes studied by Rish [32], naive Bayes trees studied by Webb *et al.,* [33], random forest studied by Breiman [34], random trees studied by Geurts *et al.,* [35], multilayer perceptron studied by Rumelhart *et al.,* [36] and support vector machines studied by Cortes and Vapnik [37] as the comparative algorithms. The inputs to these conventional machine learning algorithms will be using the same dataset with all the samples have been transformed into the 121-length features by the data preprocessing as described in Section 3. 1.

4.2 Binary Classification

This study uses a variety of performance metrics, such as accuracy, precision, recall, and F1 Score to objectively evaluate the performance of the classification models.

In this context, the term true positive (TP) refers to the number of occurrences that were successfully classified as abnormal, whereas true negative (TN) refers to the number of instances that were successfully classified as normal. The number of normal patterns that were mistakenly identified as anomalous is referred to as false positive (FP), and the number of anomalous patterns that were mistakenly identified as normal is indicated by false negative (FN).

In the binary classification experiment, normal class consists of the normal samples and abnormal class consists of all the DoS, Probe, U2R and R2L samples, Table 5 presents the results of binary classification. The proposed CNN-LSTM-SA achieved F1 scores of 92. 95% and 89.4% for identifying normal and abnormal classes, respectively and attained the highest average F1 score of 91. 17%. These results have verified that the proposed method is superior compared to the conventional benchmarked classifiers. Finally, the CNN-LSTM-SA surpassed the previously mentioned methods in terms of accuracy (refer to Figure 6), attaining the highest accuracy at 89. 36%.

Table 5
Comparison results of binary classification in term of precision, recall and F1 score using KDDTest+ testing set

Category	Proposed CNN-LSTM-SA	Decision trees	Naive Bayes	Naive Bayes trees	Random forest	Random trees	MLP	SVM
	Precision							
Normal	88.31%	85.41%	82.46%	80.93%	87.79%	81.62%	83.96%	79.95%
Abnormal	85.93%	83.05%	81.95%	79.58%	80.17%	84.24%	81.44%	82.94%
AVG	87.12%	84.23%	82.21%	80.26%	83.98%	82.93%	82.70%	81.45%
	Recall							
Normal	98.10%	78.47%	84.01%	81.79%	87.92%	86.72%	77.86%	71.25%
Abnormal	93.16%	84.45%	78.76%	87.11%	70.49%	92.83%	82.48%	89.23%
AVG	95.63%	81.46%	81.39%	84.45%	79.21%	89.78%	80.17%	80.24%
	F1 score							
Normal	92.95%	81.79%	83.23%	81.36%	87.85%	84.09%	80.80%	75.35%
Abnormal	89.40%	83.74%	80.32%	83.17%	75.02%	88.33%	81.96%	85.97%
AVG	91.17%	82.77%	81.78%	82.27%	81.44%	86.21%	81.38%	80.66%

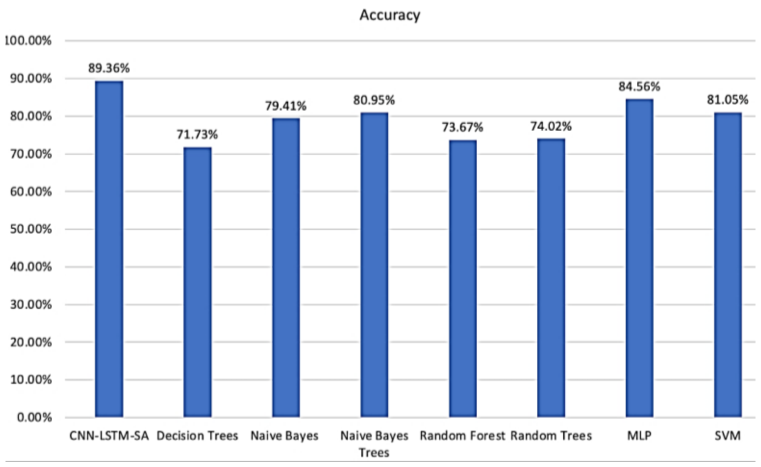

Fig. 6. Comparison results of binary classification in term of accuracy using KDDTest+ testing set

4.3 Multi-Classification

Table 6 presents the results of multi-classification experiments, the proposed CNN-LSTM-SA method surpassed all other conventional classifiers, yielding an average F1 scores as high as 93. 26%. Moreover, it is important to note that the proposed CNN-LSTM-SA method also outperformed all other conventional classifiers in terms of accuracy, achieving the highest accuracy of 93. 72% (as illustrated in Figure 7).

The recall of U2R is relatively poor due to class imbalance. Specifically, the number of instances of U2R is significantly lower than the others. The models tend to be biased towards the class with a larger number of instances, which can cause lower recall for the minority class.

Table 6
Comparison results of multi-classification in term of precision, recall and F1 score using KDDTest+ testing set

Category	Proposed CNN-LSTM-SA	Decision trees	Naive Bayes	Naive Bayes trees	Random forest	Random trees	MLP	SVM
	Precision							
Normal	87.32%	78.01%	75.87%	77.92%	73.98%	78.89%	76.45%	77.32%
Dos	99.16%	73.97%	82.49%	78.91%	94.66%	93.57%	78.26%	78.48%
Probe	83.39%	75.32%	79.01%	73.64%	78.44%	80.59%	75.47%	74.58%
R2L	97.66%	93.87%	90.36%	92.29%	93.16%	77.64%	82.46%	91.04%
U2R	91.69%	85.33%	81.91%	86.63%	89.02%	83.01%	84.29%	86.28%
AVG	91.84%	81.30%	81.93%	81.88%	85.85%	82.74%	79.39%	81.54%
	Recall							
Normal	98.87%	91.96%	86.84%	84.96%	88.27%	87.11%	90.43%	89.86%
Dos	97.82%	92.13%	92.35%	85.15%	93.49%	87.25%	89.75%	87.68%
Probe	95.56%	93.49%	87.68%	88.07%	88.91%	89.65%	84.72%	85.79%
R2L	93.68%	87.51%	85.89%	85.32%	87.84%	89.11%	89.97%	88.81%
U2R	89.16%	93.63%	90.08%	91.27%	89.99%	89.78%	94.06%	87.07%
AVG	95.02%	91.74%	88.57%	86.95%	89.70%	88.58%	89.79%	87.84%
	F1 score							
Normal	92.74%	84.41%	80.99%	81.29%	80.50%	82.80%	82.85%	83.12%
Dos	98.49%	82.06%	87.14%	81.91%	94.07%	90.30%	83.61%	82.83%
Probe	89.06%	83.43%	83.12%	80.21%	83.35%	84.88%	79.83%	79.79%
R2L	95.63%	90.58%	88.07%	88.67%	90.42%	82.98%	86.05%	89.91%
U2R	90.41%	89.29%	85.80%	88.89%	89.50%	86.26%	88.91%	86.67%
AVG	93.26%	85.95%	85.02%	84.19%	87.57%	85.44%	84.25%	84.46%

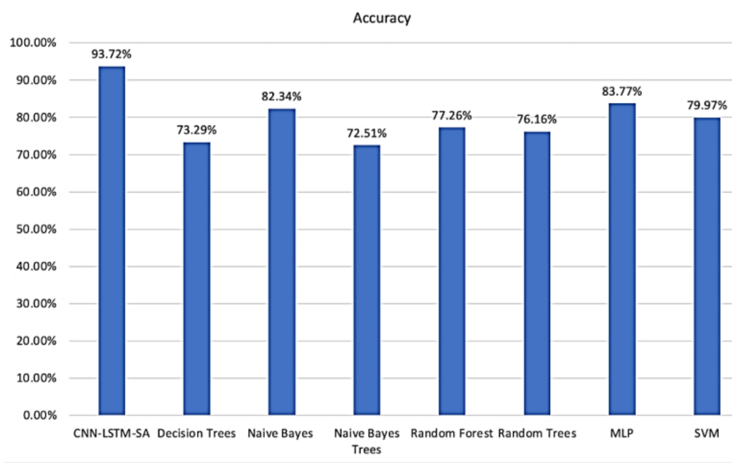

Fig. 7. Comparison results of multi-classification in term of accuracy using KDDTest+ testing set

4.4 Performance Comparison

With the same configuration, we compared the following three papers.

I. Gurung *et al.,* [38] claimed that their deep learning method achieved 87.20% accuracy and 88.51% F1 score in the binary classification evaluation using the same NSL-KDD dataset. This result is lower than the F1 score and accuracy that the proposed CNN-LSTM-SA method had achieved.
II. Shone *et al.,* [39] claimed that their deep learning method achieved 85.42% accuracy and average F1 score of 87.37% in performing the 5 classes based on the NSL-KDD dataset. This result is lower than the 93.72% accuracy and average F1 score of 93.26% that we have achieved.
III. Ieracitano *et al.,* [40] used the same configuration environment as ours, and their binary classification accuracy and multi-classification accuracy are 84.24% and 87.00%, respectively.

The comparison results of binary classification and multi-classification between the proposed method and the similar researches discussed above are tabulated in Table 7 and 8. All the values (F1 score and Accuracy) reported here the average. These results have further validated the superiority of the proposed method.

Table 7
Comparison results of binary classification for different researches

	Proposed CNN-LSTM-SA	Gurung *et al.,* [38]	Ieracitano *et al.,* [40]
F1 score	91.17%	88.51%	81.98%
Accuracy	89.36%	87.20%	84.24%

Table 8
Comparison results of multi-classification for different researches

	Proposed CNN-LSTM-SA	Shone *et al.,* [39]	Ieracitano *et al.,* [40]
Average F1 score	93.26%	87.37%	81.21%
Accuracy	93.72%	85.42%	87.00%

5. Conclusion and Future Works

In conclusion, this study has demonstrated the effectiveness of various machine learning algorithms for network intrusion detection, highlighting the superior performance of the proposed CNN-LSTM-SA method over other conventional classifiers. The proposed CNN-LSTM-SA method achieved highest F1 scores and accuracy in both binary and multi-classification experiments. By leveraging deep learning techniques, the proposed method significantly improved the detection of network intrusions, underscoring the potential of integrating advanced machine learning approaches in NID systems to strengthen cybersecurity defenses.

While the proposed CNN-LSTM-SA method has shown promising results, there remains many opportunities for further research and development in network intrusion detection. Future work may include further investigation on the applicability and performance of other advanced machine learning and deep learning techniques, such as GANs, reinforcement learning and transformer-based models in the NID systems. Another direction is to research the potential of incorporating explainable AI methods to provide insights into the decision-making processes of NID systems, increasing the transparency and trustworthiness.

Acknowledgement

The funding for this research is made possible through the Faculty of Computer Science and Information Technology, UNIMAS.

References

[1] El Omda, Mahmoud, Mohamed Helmy Megahed, and Mohamed Hassan Abdel Azeem. "Design and Simulation of New Anonymous Intelligent Authentication for 4G (LTE) Mobile Communication Network." *Journal of Advanced Research in Applied Mechanics* 41, no. 1 (2018): 1-8. https://doi.org/10.21608/iceeng.2018.30166

[2] Jusoh, WWI Wan, KA Mohd Annuar, S. H. Johari, M. H. Harun, and I. M. Saadon. "Motorcycle security system using GSM and RFID." *Journal of Advanced Research in Applied Mechanics* 16, no. 1 (2015): 1-9.

[3] Noroozi, E., S. M. Daud, and A. Sabouhi. "A Security Enhanced Robust Image Hiding Algorithm from Digital Signature." *Journal of Advanced Research in Applied Mechanics* 8, no. 1 (2015):1-12.

[4] Ibrahim, Musibau Adekunle, Patrick Ozoh, and Oladotun Ayotunde Ojo. "Fraud Detection Model for Illegal Transactions." *Journal of Computing and Social Informatics* 3, no. 1 (2024): 8-17. https://doi.org/10.33736/jcsi.6449.2024

[5] Li, Guoquan, Zheng Yan, Yulong Fu, and Hanlu Chen. "Data fusion for network intrusion detection: a review." *Security and Communication Networks* 2018 (2018). https://doi.org/10.1155/2018/8210614

[6] Chaabouni, Nadia, Mohamed Mosbah, Akka Zemmari, Cyrille Sauvignac, and Parvez Faruki. "Network intrusion detection for IoT security based on learning techniques." *IEEE Communications Surveys & Tutorials* 21, no. 3 (2019): 2671-2701. https://doi.org/10.1109/COMST.2019.2896380

[7] Shone, Nathan, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. "A deep learning approach to network intrusion detection." *IEEE transactions on emerging topics in computational intelligence* 2, no. 1 (2018): 41-50. https://doi.org/10.1109/TETCI.2017.2772792

[8] Lazarevic, Aleksandar, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. "A comparative study of anomaly detection schemes in network intrusion detection." In *Proceedings of the 2003 SIAM international conference on data mining*, pp. 25-36. Society for Industrial and Applied Mathematics, 2003. https://doi.org/10.1137/1.9781611972733.3

[9] Ghorbani, Ali A., Wei Lu, and Mahbod Tavallaee. *Network intrusion detection and prevention: concepts and techniques*. Vol. 47. Springer Science & Business Media, 2009. https://doi.org/10.1007/978-0-387-88771-5

[10] Dokas, Paul, Levent Ertoz, Vipin Kumar, Aleksandar Lazarevic, Jaideep Srivastava, and Pang-Ning Tan. "Data mining for network intrusion detection." In *Proc. NSF Workshop on Next Generation Data Mining*, pp. 21-30. Citeseer, 2002.

[11] Olofintuyi, Sunday Samuel. "A Three-Tier Model for Intrusions Classification on a Computer Network." *Journal of Computing and Social Informatics* 2, no. 2 (2023): 1-8.

[12] Anderson, James P. "Computer security threat monitoring and surveillance." *Technical Report, James P. Anderson Company* (1980).

[13] Xiao, Yihan, Cheng Xing, Taining Zhang, and Zhongkai Zhao. "An intrusion detection model based on feature reduction and convolutional neural networks." *IEEE Access* 7 (2019): 42210-42219. https://doi.org/10.1109/ACCESS.2019.2904620

[14] Sinclair, Chris, Lyn Pierce, and Sara Matzner. "An application of machine learning to network intrusion detection." In *Proceedings 15th annual computer security applications conference (ACSAC'99)*, pp. 371-377. IEEE, 1999.

[15] Vitrià, Jordi, Petia Radeva, and Isabel Aguiló, eds. "Recent Advances in Artificial Intelligence Research and Development." (2004).

[16] Kim, Kwangjo, Muhamad Erza Aminanto, and Harry Chandra Tanuwidjaja. *Network intrusion detection using deep learning: a feature learning approach*. Springer, 2018. https://doi.org/10.1007/978-981-13-1444-5

[17] Garcia-Teodoro, Pedro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. "Anomaly-based network intrusion detection: Techniques, systems and challenges." *computers & security* 28, no. 1-2 (2009): 18-28. https://doi.org/10.1016/j.cose.2008.08.003

[18] Catania, Carlos, and Carlos García Garino. "Towards reducing human effort in network intrusion detection." In *2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*, vol. 2, pp. 655-660. IEEE, 2013. https://doi.org/10.1109/IDAACS.2013.6663006

[19] Mukkamala, Srinivas, Guadalupe Janoski, and Andrew Sung. "Intrusion detection using neural networks and support vector machines." In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, vol. 2, pp. 1702-1707. IEEE, 2002.

[20] Alazab, Mamoun, Sitalakshmi Venkatraman, Paul A. Watters, and Moutaz Alazab. "Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures." *AusDM* 11 (2011): 171-182.

[21] Wang, Wei, Yiqiang Sheng, Jinlin Wang, Xuewen Zeng, Xiaozhou Ye, Yongzhong Huang, and Ming Zhu. "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection." *IEEE access* 6 (2017): 1792-1806. https://doi.org/10.1109/ACCESS.2017.2780250

[22] Vinayakumar, Ravi, Mamoun Alazab, K. Padannayil Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and Sitalakshmi Venkatraman. "Deep learning approach for intelligent intrusion detection system." *Ieee Access* 7 (2019): 41525-41550. https://doi.org/10.1109/ACCESS.2019.2895334

[23] Brown, Andy, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. "Recurrent neural network attention mechanisms for interpretable system log anomaly detection." In *Proceedings of the first workshop on machine learning for computing systems*, pp. 1-8. 2018. https://doi.org/10.1145/3217871.3217872

[24] Yin, Chuanlong, Yuefei Zhu, Jinlong Fei, and Xinzheng He. "A deep learning approach for intrusion detection using recurrent neural networks." *Ieee Access* 5 (2017): 21954-21961. https://doi.org/10.1109/ACCESS.2017.2762418

[25] Ahmad, Muhammad, Qaiser Riaz, Muhammad Zeeshan, Hasan Tahir, Syed Ali Haider, and Muhammad Safeer Khan. "Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set." *EURASIP Journal on Wireless Communications and Networking* 2021 (2021): 1-23. https://doi.org/10.1186/s13638-021-01893-8

[26] Benaddi, Hafsa, Khalil Ibrahimi, Abderrahim Benslimane, and Junaid Qadir. "A deep reinforcement learning based intrusion detection system (drl-ids) for securing wireless sensor networks and internet of things." In *Wireless Internet: 12th EAI International Conference, WiCON 2019, TaiChung, Taiwan, November 26–27, 2019, Proceedings 12*, pp. 73-87. Springer International Publishing, 2020. https://doi.org/10.1007/978-3-030-52988-8_7

[27] Cup, K. D. D. "http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html." *The UCI KDD Archive* (1999).

[28] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012).

[29] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9, no. 8 (1997): 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

[30] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

[31] Breiman, L., J. H. Friedman, R. Olshen, and C. J. Stone. "Classification and Regression Trees." (1984).

[32] Rish, Irina. "An empirical study of the naive Bayes classifier." In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, pp. 41-46. 2001.

[33] Webb, Geoffrey I., Janice R. Boughton, and Zhihai Wang. "Not so naive Bayes: aggregating one-dependence estimators." *Machine learning* 58 (2005): 5-24. https://doi.org/10.1007/s10994-005-4258-6

[34] Breiman, Leo. "Random forests." *Machine learning* 45 (2001): 5-32. https://doi.org/10.1023/A:1010933404324

[35] Geurts, Pierre, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees." *Machine learning* 63 (2006): 3-42. https://doi.org/10.1007/s10994-006-6226-1

[36] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *nature* 323, no. 6088 (1986): 533-536. https://doi.org/10.1038/323533a0

[37] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." *Machine learning* 20 (1995): 273-297. https://doi.org/10.1007/BF00994018

[38] Gurung, Sandeep, Mirnal Kanti Ghose, and Aroj Subedi. "Deep learning approach on network intrusion detection system using NSL-KDD dataset." *International Journal of Computer Network and Information Security* 11, no. 3 (2019): 8-14. https://doi.org/10.5815/ijcnis.2019.03.02

[39] Shone, Nathan, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. "A deep learning approach to network intrusion detection." *IEEE transactions on emerging topics in computational intelligence* 2, no. 1 (2018): 41-50. https://doi.org/10.1109/TETCI.2017.2772792

[40] Ieracitano, Cosimo, Ahsan Adeel, Francesco Carlo Morabito, and Amir Hussain. "A novel statistical analysis and autoencoder driven intelligent intrusion detection approach." *Neurocomputing* 387 (2020): 51-62. https://doi.org/10.1016/j.neucom.2019.11.016