Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.032107
Article

Tech Science Press

Detection of Abnormal Network Traffic Using Bidirectional Long Short-Term Memory

Nga Nguyen Thi Thanh and Quang H. Nguyen[*]

School of Information and Communication Technology, Hanoi University of Science and Technology, Hanoi, 10000, Vietnam
*Corresponding Author: Quang H. Nguyen. Email: quangnh@soict.hust.edu.vn
Received: 06 May 2022; Accepted: 28 October 2022

Abstract: Nowadays, web systems and servers are constantly at great risk from cyberattacks. This paper proposes a novel approach to detecting abnormal network traffic using a bidirectional long short-term memory (LSTM) network in combination with the ensemble learning technique. First, the binary classification module was used to detect the current abnormal flow. Then, the abnormal flows were fed into the multilayer classification module to identify the specific type of flow. In this research, a deep learning bidirectional LSTM model, in combination with the convolutional neural network and attention technique, was deployed to identify a specific attack. To solve the real-time intrusion-detecting problem, a stacking ensemble-learning model was deployed to detect abnormal intrusion before being transferred to the attack classification module. The class-weight technique was applied to overcome the data imbalance between the attack layers. The results showed that our approach gained good performance and the F1 accuracy on the CICIDS2017 data set reached 99.97%, which is higher than the results obtained in other research.

Keywords: Intrusion detection systems; abnormal network traffics; bi-directional lstm; convolutional neural network; ensemble learning

1 Introduction

The internet has become popular in society to the degree that it has led to an explosion of internet-based applications with many risks from cyberattacks targeting users and business organizations. Therefore, an intrusion detection system (IDS) is indispensable to protect systems from malware and cyberattacks. The two types of IDS are host-based intrusion detection systems (HIDS) and network intrusion detection systems (NIDS). NIDS collect and supervise network traffic using sensors deployed in multiple positions in the network. HIDSs collect and analyze the recorded information for attack detection on a specific network node. The main task of an IDS is identifying traffic by binary classification or attack type to increase the probability of detecting attacks and reducing error warnings.

An IDS is required to protect systems from malware and cyberattacks [1]. The signature-based technique in an IDS is deployed using knowledge to identify abnormal traffic. This technique has been very successful in the past, but there is a limit due to the need to update the intrusion detection knowledge system regularly.

Despite regular updates, the system is still fragile due to newly issued threats that have not been disclosed. Alternatively, traffic today is often encrypted via the SSL/TLS protocol, so it is difficult to analyze traffic content for network-based intrusion detection systems [2,3]. Many remarkable studies in the field of machine learning have tried to improve the ability to identify IDS attacks using algorithms such as SVM, K-nearest, Naive Bayes, etc., [4–8]. However, most of these algorithms used the shallow learning model, which depends mostly on the feature selection process to improve the accuracy of the algorithm. Deep learning has emerged as a solution to the problem of finding hidden features in complex data. Deep learning provides the flexibility to update new types of attacks through reliable data sets and the auto-selection feature if enough data are provided. Several studies have been done using deep learning for the problem of detecting abnormal flow, for example, Zhu et al. [7] proposed the AMF-LSTM model, and Han et al. used the reinforcement learning model. But there hasn't been any research to build a model combining traditional machine learning models with deep learning models.

In the next section, we present the related works in this field and then in the third section we propose our approach using long short-term memory (LSTM) in combination with traditional models by stacking ensemble learning. In the fourth section, we present our experiments, results, and evaluations. The last section summarizes our work and discusses our future research.

2 Related Works

The data sets published in this field include DARPA [9], KDD99 [10], NSL-KDD [11], Twente [12], ISCX2012 [13], and CICIDS2017 [4]. The DARPA data set [9] is designed for analyzing network security risks. The data set includes traffic from many protocols (e.g., email, FTP, Telnet, and SNMP) and from various attack types such as DoS, Guess Password, Buffer Overflow, remote FTP, Syn flood, Nmap, and Rootkit. However, the data set does not accurately reflect the actual traffic because it is mainly simulated in the laboratory. In addition, due to the old release date (1998), the data set does not feature new types of attacks. The KDD99 data set [10] is an updated version of the DARPA data set with the addition of new attack types such as NeptuneDoS, podDoS, and SmurfDoS. Attack and normal flows are executed in a simulated environment. However, this data set contains too much redundant and duplicate data. About 78% of the data are duplicated in the training set and 75% in the testing set, which would result in a machine learning model that focuses too much on duplicate samples and reduces the accuracy of actual data [14]. In addition, much of the traffic in the training set is also used in the testing set, resulting in the unreliability of the data set. The NSL-KDD data set [11] is inherited from the KDD99 set to solve the existing problems of the old data set. However, because it was published in 2009, the data set does not feature new forms of attacks. The Twente [12] data set contains traffic from the OpenSSH, Apache, and Proftp services obtained from a honeypot network, and therefore it is more reliable. The labeled traffic facilitates the deployment of supervised machine learning models. Nevertheless, the data set is still narrow, lacking a variety of network attack types. The ISCX2012 data set [13] is made up of two systems, the first is the Alpha system that performs the execution of attack scenarios while the Beta system performs the same tasks as the normal users. This data set includes traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP services. However, the data set does not contain the traffic of the popular HTTPS protocol. In addition, the distribution of simulated attack traffic does not closely follow reality, leading to a lack of reliability in the data set. Another data set is CICDDoS2019 [15], for which Rehman et al. [16] used a gated recurrent unit to detect and identify types of distributed denial of service (DDoS) cyberattacks with an accuracy of 99.94%. However, this data set contains only data about DDoS attacks.

One recent research area is identifying attacks against internet-connected devices. For example, Srivastava et al. [17] used the ensemble method of Linear Regression, Random Forest, and XGBoost classifiers on the UNSW-NB15 data set [18]. Imtiaz et al. [19] used a deep neural network to detect malware software in Android devices. Javed et al. [20] used an LSTM-based convolutional neural network (CNN) to detect abnormal data from sensors in vehicles.

The aforementioned data sets provide very comprehensive information about the traffic with many different types of attacks. However, due to the long-standing publication, many types of attacks are out of date and the data sets have not been updated with new attack types. However, in some data sets, because the simulated traffic is not close to reality, the data sets are less reliable. In this study, we selected the CICIDS2017 data set [4], which includes normal traffic and attacks of various services such as HTTP, HTTPS, FTP, SSH, and email protocols. From the selected data set, 80 features were extracted by the CICFlowMeter tool [21]. This data set also includes 14 common types of labeled attacks. CICIDS2017 has advantages such as fully labeled flows that facilitate a supervised learning model. The most recent and diverse attacks accurately reflect the reality of modern cyberattacks, including web attacks, brute force, DoS, DDoS, infiltration, Heartbleed, bot, and scanning. The flow has a full range of network protocols including HTTPS. The data set was created using real workstations, and network devices help to increase data reliability.

Several studies have been done on the CICIDS2017 data set. In CIC organization research, where the data set was published, Sharafaldin et al. [4] used the Random Forest regressor algorithm to extract the best characteristics for each specific type of attack. To select these characteristics, they calculated the weight of each feature corresponding to each attack. Finally, the selected features were tested for performance and accuracy with seven common machine learning algorithms. The results show that the highest F1 index reaches 98% with the ID3 model. Ahmim et al. [5] proposed a new model with different classification methods based on a decision tree and rule set, including REP Tree, JRip Algorithm, and Forest PA. In particular, the first two methods take input data as network data set characteristics and classify the information into normal and abnormal groups. The third method uses the characteristics of the original data and combines it with the data processed by two previous methods to identify each specific type of attack. This method has achieved an identification rate of up to 94.475% and has reduced lowest false alarm rate to 1,145%, which is superior to conventional machine learning algorithms such as Naive Bayes, Random Forest, SVM, etc. However, this is still a simple machine learning model with few representation layers, so not all of the data's hidden characteristics have been taken. Aksu et al. [6] proposed a model using the Fisher score algorithm to select 30 optimal characteristics from 80 statistical characteristics of the flow. Then, these optimal characteristics were applied to machine learning algorithms such as KNN, SVM, and Decision Tree. The resulting measured F1 was up to 0.99 for the DDoS data set. The advantage of this research is that it has greatly reduced unnecessary features that help machine learning algorithms to reduce training time for the model and increase accuracy significantly. Nevertheless, by applying only to the DDoS data set, the overall problem was not reflected accurately. In addition, the machine learning model they used is shallow, so the model's accuracy also depends on the specific selection of the algorithm.

The conventional IDS models detect cyberattacks based on knowledge or sets of preconfigured laws. In recent years, with the development of hardware and an abundant amount of data, deep learning has emerged as an integrated solution with IDSs to increase the accuracy of the identification.

Zhu et al. [7] proposed the AMF-LSTM model to identify abnormal flow. The model uses multiple flows instead of a training flow for the AMF-LSTM network to gain correlation between flows, and it adds an attention technique to LSTM to help the model learn which traffic has a greater influence on the result. The advantage of this method is that by taking advantage of the correlation between the traffic, the probability of detecting attacks such as DoS, DDoS, and PortScan is increased because these attacks often involve multiple traffic flows. Furthermore, the integration of the attention layer helps the model focus on the important traffic. However, the model still has some disadvantages, such as using unidirectional LSTM, so it cannot take advantage of the backward flow to make a more reasonable judgment. The model also cannot handle the data imbalance, which causes it to learn mainly on the majority of data layers such as Normal, DDOS, and PortScan. However, applying the characteristics of traffic directly into the LSTM model makes it difficult to use the feature effectively.

Han et al. [22] performed the most recent study. They used proximal policy optimization as a reinforcement learning model that trains a deep neural network as a feature extractor and uses a k-means clustering module to classify attack types. These methods were conducted in CICIDS2017 and UNSW-NB15 data sets with F1 scores of 0.96552 and 0.94268.

However, the above studies contain unresolved problems: Backward flows were not used in the machine learning models, and none of the studies combined traditional machine learning models such as KNN and Random Forest with a neural network to increase attack type recognition accuracy. Another problem is that often the data of each type of attack are not evenly distributed. In the next section, we propose a new method that can handle these problems.

3 Proposed Abnormal Network Traffic Detection Method

3.1 The CICIDS2017 Dataset

The CICIDS2017 data set contains the normal and abnormal traffic of recent common attacks. The traffic is stored in PCAP format and the data set includes statistical features regarding traffic flows (80 extracted features from the CICFlowMeter tool [18]). Compared with the features extracted from the previous data set (KDD99, NSL, etc.), CIC has removed the statistical features from the previous traffic in the 2-second time window, such as the number of connections of the same service and the same server. This approach is reasonable because many slow attack types exist, such as the Slowloris DoS attack, in which attacking traffic lasts more than 2 s. These characteristics are also listed in both the arrival and departure directions. This data set contains traffic from 25 users on HTTP, HTTPS, FTP, SSH, and email protocols, measured from 9:00 am June 3, 2017, to 5:00 pm June 7, 2017. Table 1 describes the distribution of traffic by attack type.

Table 1: Flow statistics by attack type

No.	Type of flow	Quantity	No.	Type of flow	Quantity
1	Heartbleed	11	9	SSH Patator	5,897
2	SQL Injection	21	10	FTP Patator	7,938
3	Infiltration	36	11	Dos GoldenEye	10,293
4	XSS	652	12	DDos	41,835
5	Brute Force	1,507	13	PortScan	158,930
6	Bot	1,966	14	Dos Hulk	231,073
7	Dos Slowhttptest	5,499	15	Benign	2,359,289
8	Dos Slowloris	5,796			

The data set contains 288,602 unlabeled flows and 203 missed specific information flows. All of these flows will be removed from the training set, resulting in 2,830,540 remaining flows. In addition, the quantity of flow in each layer of the data set is seriously imbalanced. The data are mainly in the benign class, and the other classes have a smaller amount of data, especially the Heartbleed, SQL Injection, and Infiltration attack types. Therefore, our proposed model will need to pay attention to these types of attacks.

The overall architecture of the system consists of two main blocks (Fig. 1):

- The preprocessing block performs feature extraction from the network traffic file using the CICFlowMeter tool, each flow has 80 features [18]. The tool cleans the data, removes redundant data, and divides each 40-flow block data into three volumes. The tool calculates the mean, maximum, and minimum values of the training set to rescale all of the training, validating, and test sets.
- The train and test blocks include a stacking model for binary classification in combination with the proposed model of multilayered attack classification.

Details of the components in the system will be described in the below sections.

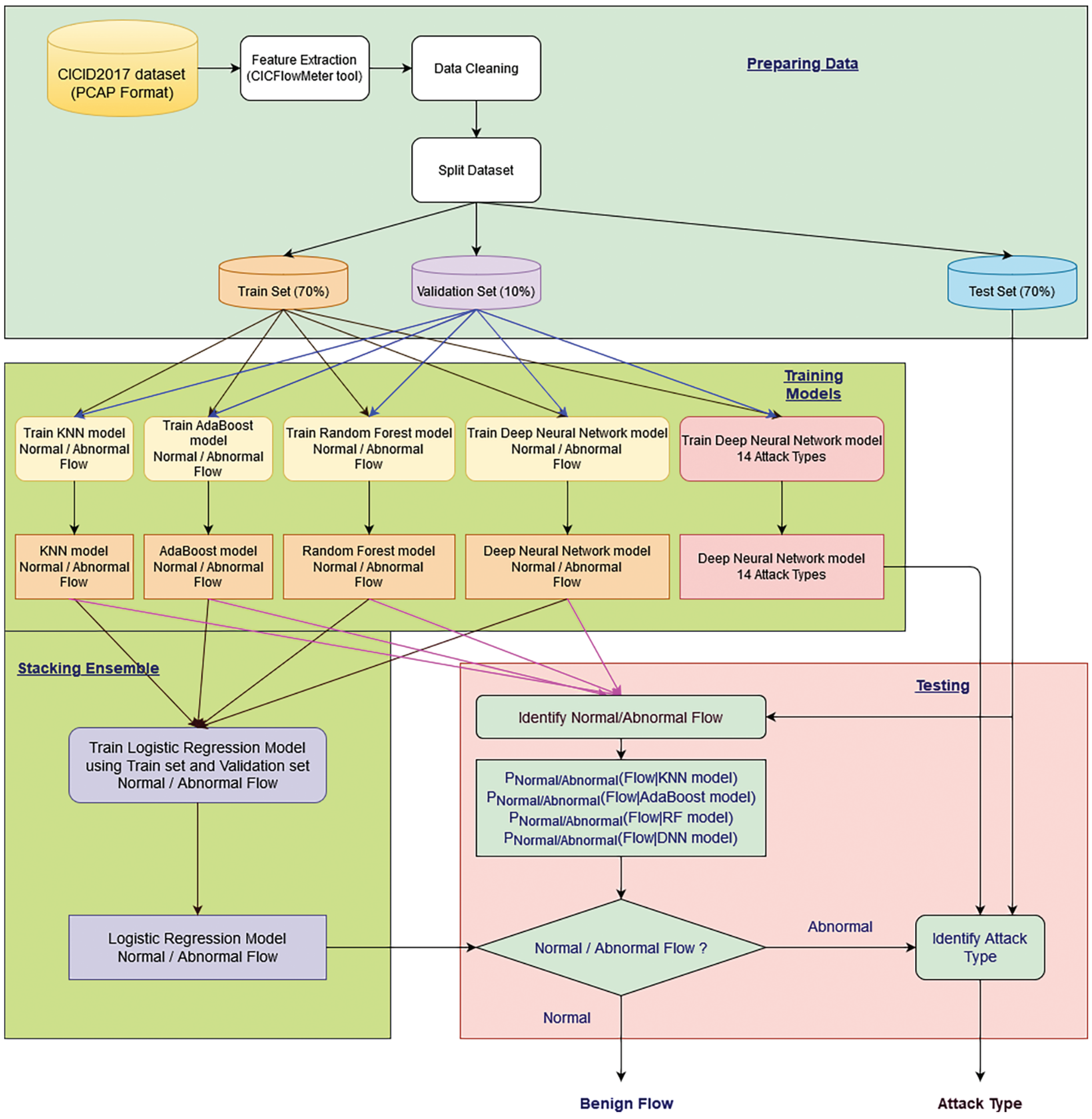
CICID2017 dataset (PCAP Format)
Feature Extraction (CICFlowMeter tool)
Data Cleaning
Preparing Data
Split Dataset
Train Set (70%)
Validation Set (10%)
Test Set (70%)
Training Models
Train KNN model Normal / Abnormal Flow
Train AdaBoost model Normal / Abnormal Flow
Train Random Forest model Normal / Abnormal Flow
Train Deep Neural Network model Normal / Abnormal Flow
Train Deep Neural Network model 14 Attack Types
KNN model Normal / Abnormal Flow
AdaBoost model Normal / Abnormal Flow
Random Forest model Normal / Abnormal Flow
Deep Neural Network model Normal / Abnormal Flow
Deep Neural Network model 14 Attack Types
Stacking Ensemble
Testing
Identify Normal/Abnormal Flow
$P_{Normal/Abnormal}$(Flow|KNN model)
$P_{Normal/Abnormal}$(Flow|AdaBoost model)
$P_{Normal/Abnormal}$(Flow|RF model)
$P_{Normal/Abnormal}$(Flow|DNN model)
Train Logistic Regression Model using Train set and Validation set Normal / Abnormal Flow
Logistic Regression Model Normal / Abnormal Flow
Normal / Abnormal Flow ?
Abnormal
Identify Attack Type
Normal
Benign Flow
Attack Type

Figure 1: The overall architecture of the proposed abnormal network traffics detection model

3.2 Preprocessing Data

As stated in the overview of the CICIDS2017 data set, the data set had many limitations before it could be put into the training model. Preprocessing data was divided into four parts: (a) data cleaning, (b) one-hot coding for strings, (c) normalization technique with numerical features, and (d) splitting training, validating, and testing the set by timeline.

3.2.1 Data Cleaning

The data set contains 288,602 unlabeled and 203 missed for specific information flows. All of the flows were removed from the training set, so the remaining data set contained 2,830,540 flows. We eliminated values with zero variance to increase the accuracy of the model. This elimination is reasonable because, in theory, zero variance values do not contain any useful information for attack classification. The data set contains the features obtained from flows, and they are related to the characteristics of the computer network and do not affect the model results. Adding these features can easily lead to noise and orient the judgment models according to the IP or ID of the flow. At this step, we do not need to process data according to feature selection techniques because, with a sufficient amount of data, the deep learning model can select on its own the important characteristics that affect the anticipated results during the training process. This advantage is also a point of superiority of deep learning compared to other machine learning algorithms. After cleaning the data, only 70 features were retained to train the model.

3.2.2 One-Hot Coding for Strings

The deep learning model requires input data to be a numerical value. In the above-mentioned features, labels are in string form. Therefore, labels needed to be converted into digital vectors by a one-hot coding technique. This encoding format is compatible with the softmax output layer of the model. Each sample will produce a prediction of 14 probability values corresponding to 14 attack types. The valid element position is an instance of the class's position.

3.2.3 Normalization Technique with Numerical Feature

In a neural network, when the input data attributes are in different ranges, the loss function is narrow in the weights **w** and **b** coordinate system, thus causing the gradient descent algorithm (network training algorithm) to have to take additional steps to reach the optimal point. To bring the data set to the same value range [−1,1], we applied the max-min normalization technique in Eq. (1):

$$z = \frac{x - mean}{max - min} \tag{1}$$

where x is the input data, z is the value after performing normalized input, *mean* is the mean value of the feature, *max* is the maximum value of the feature, and *min* is the minimum value of a feature. The *mean*, *max*, and *min* values are computed with the data in the training set, and then applied to the validating and test sets.

3.2.4 Splitting Training, Validating, and Test Set by Timeline

We organized the input data in the form of a 2D matrix: time step and characteristic vector. The input data set was divided into groups, with each group containing 40 flows. To ensure the timeline continuity of the training flows, we separated them into the training, testing, and validation sets in blocks of 40 consecutive flows. The sequence number of each block was set to be an ID, and a partitioning algorithm was used to ensure the random division and the ratio of 80:10:10 for each layer. The data distribution after dividing blocks according to the attack's type is described in Table 2.

Table 2: Flow block statistics by attack type

Attack type	Train set	Validation set	Test set
Benign	1,820,102	223,031	228,153
Bot	1,577	195	184
DDos	101,862	13,312	12,851
Dos GoldenEye	8,142	1,194	955
Dos Hulk	185,228	22,893	22,003
Dos Slowhttptest	4,502	597	400
Dos Slowloris	4,978	444	374
FTP Patator	6,331	758	846
Heartbleed	7	3	1
Infiltration	26	5	5
PortScan	125,893	16,757	16,154
SSH Patator	4,654	580	663
Brute Force	1,206	135	166
SQL injection	17	1	3
XSS	555	55	42

3.3 Model Overview

The IDS model architecture is proposed in Fig. 1. At first, the binary classification module was used to detect the current flow as normal or abnormal. In the abnormal flow case, the flow was transferred to the multilayer classification module to identify the specific type of attack.

3.3.1 Binary Classification Module

Ensemble learning is a technique that combines multiple predictive models to produce better results than each model would alone. We used this technique by combining the basic deep learning model (binary classification model, presented later) with other machine learning models to identify normal and abnormal flows. Because this model is unidirectional, it needs enough sets of 10 flows to be able to identify traffic, thereby guaranteeing real-time processing. In the normal flow case, the flows are forwarded to the users; otherwise, they are moved to our proposed bidirectional model with the next 10 flows to identify the specific type of attack. The ensemble learning technique is used to combine the basic deep learning model with Random Forest, AdaBoost, and KNN. This technique is based on the assumption that each model will see the data set from a different perspective to make its predictions, which means each model would find part of the "truth" but not all. Based on this characteristic, we can completely synthesize various perspectives from various machine learning models to make a much more accurate prediction. The structural diversity of machine learning models has a high effect on the ensemble learning technique in which various models and branches are combined, not only focusing on neural network models but also on other machine learning models such as Random Forest, AdaBoost, or K-nearest neighbor. The basic method is the weighted combination. In this method, each model makes predictions for the validation set. The predicted results for each model are multiplied by the weights a1, a2, a3, etc, and then they are added up (Eq. (2)). The sum of these weights equals 1. The weight selections will be based on the most satisfactory final prediction. Finally, predictions for the test set are made to evaluate the model results.

$$P_{Normal/Abnormal}(FlowX) = a_1 * P_1 + a_2 * P_2 + a_3 * P_3 + a_4 * P_4 \tag{2}$$

In Eq. (2):

- $P_1 = P_{Normal/Abnormal}(FlowX|KNNmodel)$
- $P_2 = P_{Normal/Abnormal}(FlowX|AdaBoostmodel)$
- $P_3 = P_{Normal/Abnormal}(FlowX|RandomForestmodel)$
- $P_4 = P_{Normal/Abnormal}(FlowX|DNNmodel)$

The advantage of this method is that it is quick and effective. However, the method is not flexible enough and it is difficult to find an appropriate set of weights. To address the above disadvantages, the ensemble stacking technique introduces a machine learning model such as logistic regression to learn the weighted indexes automatically. This method is similar to the weighted-combination technique above, but it is combined with the logistic regression model to learn weights automatically from the predictions. The input of this model is the predicted y-predict values of many models and the output is the actual y value. The ensemble model learns how to match the actual y value and it is trained on a training set. The models were trained on a validation set and experimented on the test set to evaluate the results. We used 4 models including deep learning, Random Forest, AdaBoosting, K-nearest-neighbor in combination to make the predictions.

3.3.2 The Deep Learning Model for Normal/Abnormal Flow Detection

The deep learning model architecture is described in Fig. 3. The input data for the model is a sequence of 10 previous flows, including the predicted flow from every 70 features extracted. Therefore, the input block will be in the form (10,70). The next step is processing time-series data by LSTM (Fig. 2) with 256 output units for the output block of (10,256). The outputs of the LSTM layer will be duplicated into two branches. One branch will enter the attention block (upper branch) to calculate the weight of the attention index for each time step through the softmax class and then multiply back to the other branch. Finally, the layer is fully connected to the sigmoid activation function with an output to identify normal or abnormal traffic.

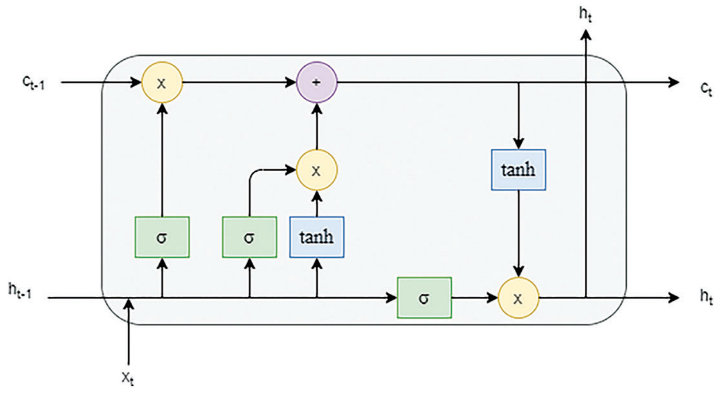

Figure 2: Long-short tem memory neural network

3.3.3 Deep Learning Model for Classification of 14 Attack Types

The architecture is described in Fig. 4. In this model, the model input is a sequence of flows containing 70 features in each flow. The model output is a prediction from the 14 attack types listed in the previous section. For each flow that needs to be predicted, a sequence of 10 previous flows and 10 following flows was put into the model. This input sequence was passed through the lambda layer for separation into two chains: the first and the next 10 flow sequences. The two sequences after separation were processed through the CNN1D blocks and the LSTM in combination with the attention technique. Finally, the results were combined through the concatenate class. The output layer was the layer fully connected with the Softmax function to make predictions. The final layer (output layer) of the model before giving the result was the fully connected Softmax layer with 14 output values (corresponding to the 14 types of attacks).

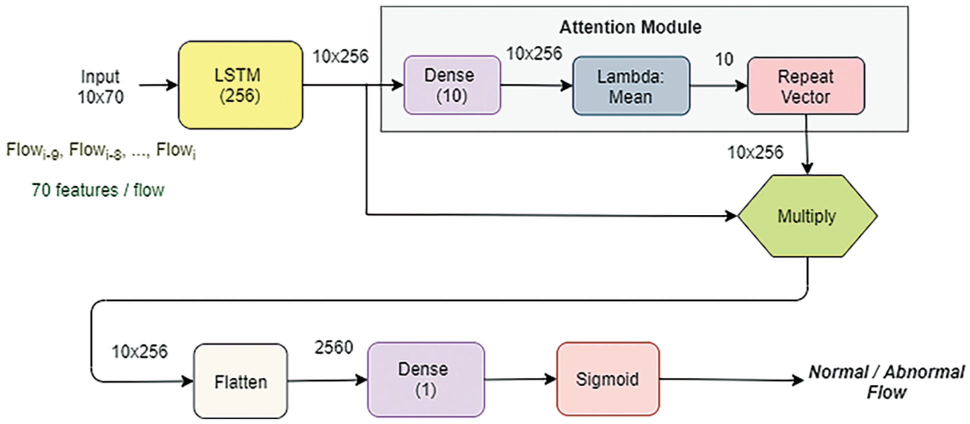

Figure 3: Basic deep learning model architecture to classify traffic as normal or abnormal flows

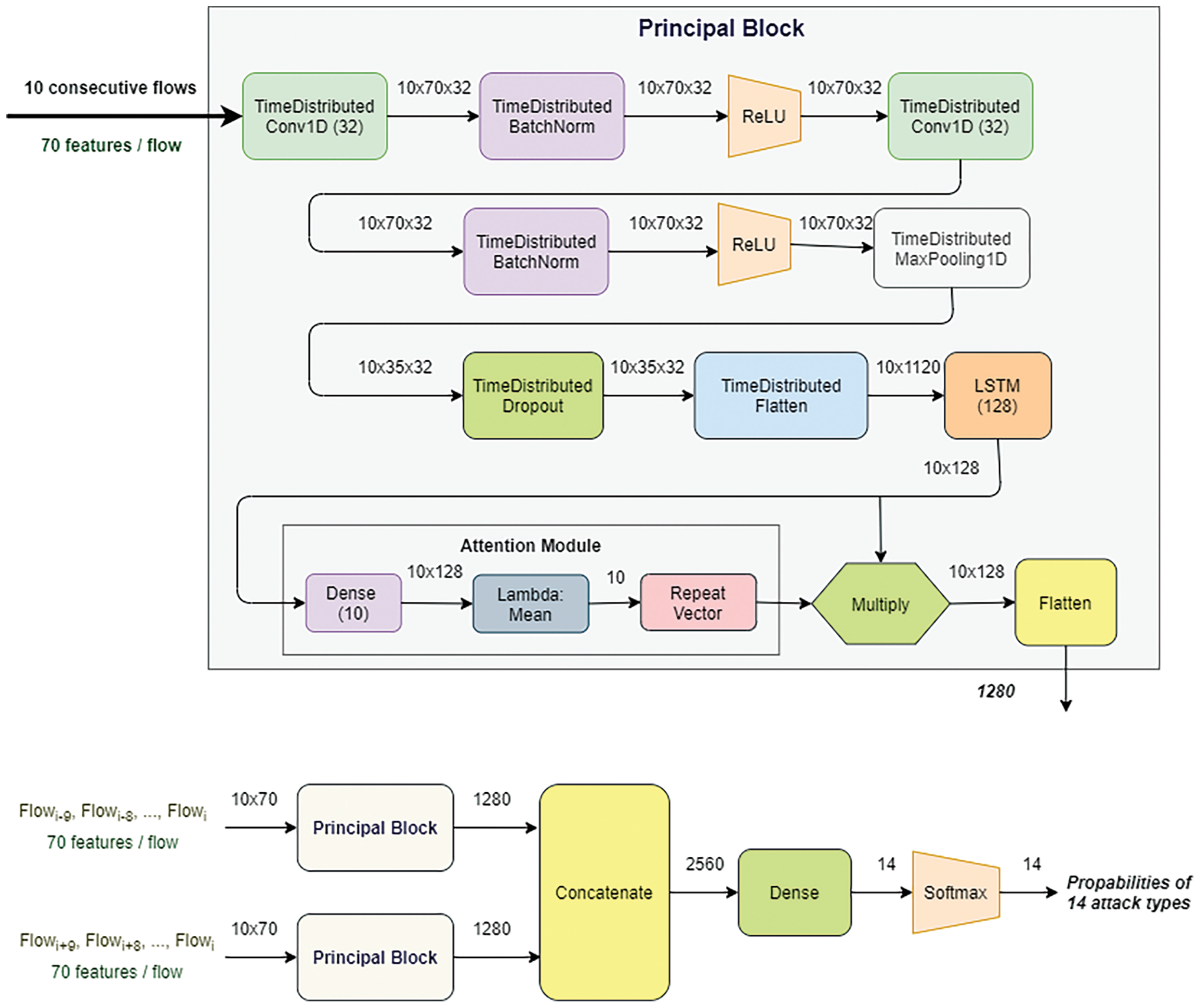

Figure 4: Proposed model architecture to detect IDS attack type

In both Deep Learning models, we use the CNN 1D layer before putting data into LSTM. CNN network [23] will extract input features into more optimal features, and find out hidden attributes before putting them into LSTM [21], thereby reducing training time and increasing the accuracy of the model. In addition, as discussed in the first section, the types of attacks such as Dos, DDos, Port Scan, and Patator are related to many traffics in an attack. Therefore, the information obtained from adjacent traffic to predict a specific flow are essential and will increase the probability of accurate prediction significantly. Because many memory steps are put into a flow sequence, the basic regression network model is difficult to handle because of the derivative problem with a long memory step. Then, the LSTM model accepts longer processing time but more accurate predictions are used. The usage of the bi-direction model is to overcome the limitations of the unidirectional LSTM network, providing more information about the traffic behind the need-to-be-predicted flow will be a great advantage when making judgments.

3.3.4 Processing Techniques with Unbalanced Data Sets, Focusing on the Optimal Accuracy of the Model in the Bot, Brute Force Minority Groups

Many techniques exist for dealing with unbalanced data sets, such as undersampling, oversampling, and weighted class. The undersampling technique selects only a few elements of the dominant class and combines them with the other classes to make training data. The advantage of this technique is that by eliminating some samples of the dominant class, the training model is faster and better able to focus on the minority class, but the downside is the loss of information and the reduced accuracy of the dominant class. Meanwhile, the oversampling technique repeats the data or combines the data to create new data for the class with less data and combines the information with the other class to make training data. This technique helps to improve focus on the minority class, but due to newly generated data, the potential for interference that reduces the accuracy of the minority class or bursts data exists if minority classes make up a small percentage.

In this research, the weighted class technique was used. This technique rates a high-weighted index for minority class samples in the loss function to reduce the weight update slope. This process helped the algorithm to focus optimally on predicting the samples of the minority class such as Bot and Brute Force. Then, the two classes were given a greater weight of 3, and the remaining classes were weighted by 1.

3.3.5 The Technique of Selecting Model Optimal Hyper-parameters in Parameter Grid Space by Parzen Ttree Algorithm

In a neural network, there are several hyper-parameters, which are the parameters of the algorithm such as the number of classes, number of units per class, and loss function. These parameters are initialized, unchanged during the training process, and not affected by optimization algorithms such as gradient descent (different from the model parameters such as **W**, **b** which are optimized through each epoch during the training process via back-propagation algorithm). Optimizing **W** and **b** for the model is very simple because they have functional dependencies on the loss function and we only need to run the gradient descent algorithm to determine their change direction. In the case of hyper-parameters, these parameters exist in discrete space without any function dependency on the loss function, therefore, the direction of parameters needs to be decided without using the derivative.

Usually, the optimization of the hyper-parameters depends largely on the experiences of the researcher. There has been a lot of research about the automated selection of this hyper-parameter. The essence of these techniques, by evaluating the effectiveness of the model with the previous validation set, is selecting the next set of hyper-parameters based on Bayes, random search, and Parzen trees techniques. In this study, we use the Hyperas library to optimize these parameters.

4 Experiments, Results, and Discussions

4.1 Performance Metrics

For a multi-layered classification problem with a severely unbalanced dataset, the Precision, Recall, and F1 Score measurements are the most reasonable. The three most important metrics are Precision (Eq. (3)), Recall (Eq. (4)), and F1 (Eq. (5)).

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{5}$$

4.2 Experiment Results and Discussion

The program was built using the Python language based on the Keras toolkit with Scikit-learn on Linux Ubuntu 18.04. The experiments were performed on computers with Intel Core i7 CPU configuration, 128 GB RAM, and NVIDIA 2080TI GPU.

The loss function graph of the training and validation sets during the training process is described in Fig. 5. We found that the model converged very quickly at the first five epochs where the loss function value of both the training and validation sets dropped sharply below 0.025. However, from the 9th epoch, although the loss function value of the training set still decreased steadily, it did not in the validation set and even showed signs of increasing. This is the overfitting phenomenon occurring after the 9th epoch. Table 3 describes the detailed results of each attack type.

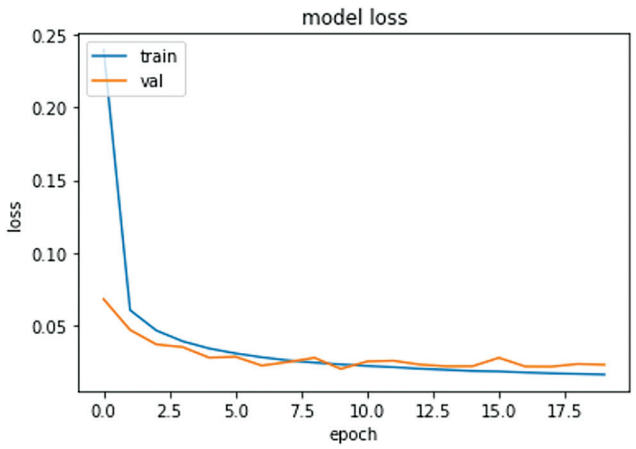

Figure 5: Loss function graph of the training and validation sets during the training process

Table 3 describes the summary results of our tests. Based on the results, we have the following comments:

- Infiltration, SQL injection, and XSS attacks have poor results because the data related to these attacks in the data set are very small (< 50 samples in the entire data set).
- Therefore, we focused on attack types that have a small number of samples but are likely to appear as Bot and Brute Force (1,966 samples and 1,507 samples in the entire data set). The results in Table 4

show that the system effectively detected these attacks, with F1 scores of 0.79 and 0.78, respectively. Otherwise, Zhu et al. [7] reported that F1 score for detecting Bot attack was only 0.1 while other studies have barely published results on these types of attacks.

- The system is capable of detecting the remaining types of attacks very well, with F1 ≥ 0.96.
- In addition, our proposed deep learning, bidirectional LSTM model combined with ensemble learning had the best results with an average F1 of 99.79%.

Table 3: F1 results of abnormal network traffic attack classes

Attack type	Precision	Recall	F1-score
Benign	1.00	1.00	1.00
Bot	0.88	0.72	0.79
DDos	1.00	0.98	0.98
Dos GoldenEye	0.99	1.00	0.99
Dos Hulk	1.00	1.00	1.00
Dos Slowhttptest	1.00	0.92	0.96
Dos Slowloris	0.98	1.00	0.99
FTP Patator	1.00	1.00	1.00
Heartbleed	1.00	1.00	1.00
Infiltration	0.00	0.00	0.00
PortScan	1.00	0.99	0.99
SSH Patator	0.99	0.96	0.97
Brute Force	0.84	0.73	0.78
SQL injection	0.00	0.00	0.00
XSS	0.46	0.26	0.33

Table 4: Summary of F1 values with various models

Model name	F1
Zhu et al. [7]	0.9228
Han [22]	0.9356
Ahmim et al. [5]	0.9448
Sharafaldin et al. [4]	0.9800
Our proposal method	*0.9977*

The results in Table 4 show that our method produced higher results than other state-of-the-art research on the same CICIDS2017 data set did. In comparison with the other studies using deep learning models, Zhu et al. [7] proposed the AMF-LSTM model to identify abnormal flow and Han et al. [22] used a deep neural network as a feature extractor and uses a k-means clustering module to classify attack types, we used a deep learning model based on the LSTM model to learn hidden features and detect abnormal flow and we also used KNN, AdaBoost, and Random Forest models in combination with the deep learning model to increase performance, so our result is better than Zhu et al. [7]. To classify attack types, our proposal model used both forward and backward flows, so our result is better than Han et al. [22]. Our results are also better than the result of studies using traditional machine models as the results of Sharafaldin et al. [4] and Ahmim et al. [5]. This outcome can be explained by our method having the following main advantages:

- The method solves two problems: The first problem is detecting whether a flow is normal or abnormal, and the second problem is correctly identifying (classifying) an abnormal flow in any common type of attack.
- The proposed stacking ensemble method allows the use of a combination of several machine learning models to detect anomalous flows.
- To solve the second problem, the method integrates both forward and backward flows into the neural network architecture based on the LSTM model.
- By using the weighted class technique, the proposed method improves the accuracy of identifying attack types with a small number of training samples.

5 Conclusions

This paper presents a new method of detecting abnormal IDS queries using bidirectional LSTM in combination with ensemble learning techniques. BiLSTM is costly because it has double LSTM cells, but the bidirectional LSTM deep learning model integrated with CNN1D and an attention technique has shown its effectiveness in detecting attack types. In addition, the stacking model for abnormal intrusion detection before being introduced into the proposed model provides results similar to the proposed model and solves the real-time intrusion detection problem. The class-weight technique to solve the data imbalance among classes showed efficiency with minority attack classes (Bot and Brute Force attacks). This method has been tested on the CICIDS2017 dataset with an overall F1 result of 0.9977, where the F1 of the tests detecting attacks with small dataset Bot and Brute Attack are 0.79 and 0.78, respectively.

However, this method is constructed based on BiLSTM model, so it is still slow. In the future, we will integrate the model into the actual IDS system, further researching techniques to reduce identification time to correspond with the requirements of real-time IDSs, exploring more complex deep learning models such as ResNet and DenseNet, and testing their effectiveness. In this study, we only solved the problem of IDS attacking common websites and servers. A further research direction is to solve the IDS problem for mobile devices such as smartphones or self-driving cars.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] R. Singh, H. Kumar, R. K. Singla and R. R. Ketti, “Internet attacks and intrusion detection system: A review of the literature,” *Online Information Review*, vol. 41, no. 2, pp. 171–184, 2017.

[2] R. Hofstede, M. Jonker, A. Sperotto and A. Pras, “Flow-based web application brute-force attack and compromise detection,” *Journal of Network and Systems Management*, vol. 25, no. 4, pp. 735–758, 2017.

[3] P. Bedi, N. Gupta and V. Jindal, “I-SiamIDS: An improved Siam-IDS for handling class imbalance in network-based intrusion detection systems,” *Applied Intelligence*, vol. 51, no. 2, pp. 1133–1151, 2021.

[4] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in *Proc. ICISSP*, Funchal-Madeira, Portugal, pp. 108–116, 2018.

[5] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour and H. Janicke, “A novel hierarchical intrusion detection system based on decision tree and rules-based models,” in *Proc. DCOSS*, Santorini Island, Greece, pp. 228–233, 2019.

[6] D. Aksu, S. Üstebay, M. A. Aydin and T. Atmaca, “Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm,” in *Proc. ISCIS*, Poznan, Poland, Cham, Springer, pp. 141–149, 2018.

[7] M. Zhu, K. Ye, Y. Wang and C. Z. Xu, “A deep learning approach for network anomaly detection based on AMF-LSTM,” in *Proc. NPC*, Muroran, Japan, pp. 137–141, 2018.

[8] K. Wu, Z. Chen and W. Li, “A novel intrusion detection model for a massive network using convolutional neural networks,” *IEEE Access*, vol. 6, no. 1, pp. 50850–50859, 2018.

[9] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall *et al.,* “Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation,” in *Proc. DISCEX*, South Carolina, USA, pp. 12–26, 2000.

[10] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *CISDA*, Ottawa, Canada, pp. 1–6, 2009.

[11] M. Tavallaee, N. Stakhanova and A. A. Ghorbani, “Toward credible evaluation of anomaly-based intrusion-detection methods,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 40, no. 5, pp. 516–524, 2010.

[12] A. Sperotto, R. Sadre, F. Vliet and A. Pras, “A labeled data set for flow-based intrusion detection,” in *Proc. IPOM09*, Venice, Italy, pp. 39–50, 2009.

[13] A. Shiravi, H. Shiravi, M. Tavallaee and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” *Computers and Security*, vol. 31, no. 3, pp. 357–374, 2012.

[14] J. McHugh, “Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory,” *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, 2000.

[15] I. Sharafaldin, A. H. Lashkari, S. Hakak and A. A. Ghorbani, “Developing realistic distributed denial of service attack dataset and taxonomy,” in *Proc. ICCST*, Chennai, India, pp. 1–8, 2019.

[16] S. Rehman, M. Khaliq, S. I. Imtiaz, A. Rasool, M. Shafiq *et al.,* “DIDDOS: an approach for detection and identification of distributed denial of service cyberattacks using gated recurrent units,” *Future Generation Computer Systems*, vol. 118, no. 2, pp. 453–466, 2021.

[17] G. Srivastava, T. R. Gadekallu, N. Deepa, B. Prabadevi and P. K. Reddy, “An ensemble model for intrusion detection in the internet of softwarized things,” in *Proc. ICDCN*, Nara, Japan, pp. 25–30, 2021.

[18] M. Sarhan, S. Layeghy, N. Moustafa and M. Portmann, “NetFlow datasets for machine learning-based network intrusion detection systems,” in *Proc. BDTA*, Virtual Event, pp. 117–135, 2020.

[19] S. I. Imtiaz, S. Rehman, A. R. Javed, Z. Jalil, X. Liu *et al.,* “DeepAMD, detection and identification of android malware using high-efficient deep artificial neural network,” *Future Generation Computer Systems*, vol. 115, no. 5, pp. 844–856, 2021.

[20] A. R. Javed, M. Usman, S. U. Rehman, M. U. Khan and M. S. Haghighi, “Anomaly detection in automated vehicles using multistage attention-based convolutional neural network,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4291–4300, 2020.

[21] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] H. Han, H. Kim and Y. Kim, “An efficient hyperparameter control method for a network intrusion detection system based on proximal policy optimization,” *Symmetry*, vol. 14, no. 1, pp. 1–15, 2022.

[23] A. Krizhevsky, I. Sutskeve and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proc. NIPS*, Nevada, USA, pp. 1097–1105, 2012.