Advances in Science, Technology and Engineering Systems Journal Vol. 8, No. 6, 144-154 (2023)

www.astesj.com

Special Issue on Innovation in Computing, Engineering Science & Technology

ASTESJ

ISSN: 2415-6698

Optimizing the Performance of Network Anomaly Detection Using Bidirectional Long Short-Term Memory (Bi-LSTM) and Over-sampling for Imbalance Network Traffic Data

Toya Acharya[*], Annamalai Annamalai, Mohamed F Chouikha

Electrical and Computer Engineering, Prairie View A & M University, Prairie View, Texas,77446, USA

A R T I C L E   I N F O

Article history:
Received: 07 November, 2023
Accepted: 22 December, 2023
Online: 30 December, 2023

Keywords:
Network Anomaly Detection
Sampling
Machine Learning
Deep learning
Bidirectional-LSTM
NSL-KDD
Random Under Sampling (RUS)
Random Over Sampling (ROS)
SMOTE
Data Imbalance

A B S T R A C T

Cybercriminal exploits integrity, confidentiality, and availability of information resources. Cyberattacks are typically invisible to the naked eye, even though they target a wide range of our digital assets, such as internet-connected smart devices, computers, and networking devices. Implementing network anomaly detection proves to be an effective method for identifying these malicious activities. The traditional anomaly detection model cannot detect zero-day attacks. Hence, the implementation of the artificial intelligence method overcomes those problems. A specialized model, known as a recurrent neural network (RNN), is specifically crafted to identify and utilize sequential data patterns to forecast upcoming scenarios. The random selection of hyperparameters does not provide an efficient result for the selected dataset. We examined seven distinct optimizers: Nadam, Adam, RMSprop, Adamax, SGD, Adagrad, and Ftrl, with variations in values of batch size, epochs, and the data split ratio. Our goal is to optimize the performance of the bidirectional long short-term memory (Bi-LSTM) anomaly detection model. This optimization resulted in an exceptional network anomaly detection accuracy of 98.52% on the binary NSL-KDD dataset. Sampling techniques deal with the data imbalance problem. Random under-sampling, which involved removing data from the majority classes to create a smaller dataset, was less efficient for deep learning models. In contrast, the Synthetic Minority Oversampling Technique (SMOTE) successfully generated random data related to the minority class, resulting in a balanced NSL-KDD multiclass dataset with 99.83% Bi-LSTM model detection accuracy. Our analysis discovered that our Bidirectional LSTM anomaly detection model outperformed existing anomaly detection models compared to the performance metrics, including precision, f1-score, and accuracy.

1. Introduction

This paper is an extension of "Efficacy of Bidirectional LSTM Model for Network-Based Anomaly Detection" [1] presented at the conference IEEE 13th Symposium on Computer Applications & Industrial Electronics (ISCAIE) in 2023.

Information technology has revolutionized how essential data is conveyed, utilizing bits to transfer a wide range of information from one point to another. This transmitted data can encompass diverse forms, such as voice, images, or data, including sensitive details like banking information, personal records, and network traffic. Numerous tools and techniques are available to identify and thwart unauthorized access.

Anomaly is an unusual pattern present in the dataset. Some techniques or methods are required to detect those anomalies from the dataset. The anomaly is also called the outliers during the study of anomaly detection. Anomaly detection is used in large fields to sense abnormal patterns, such as in business, network attack detection, monitoring health conditions, detecting fraud credit card transactions, and detecting malicious activities in mission-critical systems. Detecting anomalies is critical in cyber security for achieving solid safeguards against cyber criminals. Figure 1. provides brief information about the taxonomy of anomaly detection methods [2].

The security of information resources is ensured when the three fundamental principles of computer security—confidentiality, integrity, and availability (CIA)—are appropriately obeyed [3]. An intrusion detection system is a mechanism used to monitor and scrutinize computer or network- related activities to identify potential threats by assessing the frequency at which computer security guidelines are violated based on confidentiality, integrity, and availability. Intrusion is any unwelcome and illegal activity within an organization's internet-connected end terminal or

[*]Corresponding Author: Toya Acharya, tacharya@pvamu.edu

network-connected devices. These illegal activities aim to gain entry to a business computer or network device. An alternative term for intrusion is a malicious activity that disrupts the fundamental principles of information resource protection known as the CIA triad. An intrusion detection system examines computer network and host activities, pinpointing dubious traffic and abnormalities. Intrusion detection and prevention systems scrutinize traffic from both internal and external sources to identify potentially malicious actions.

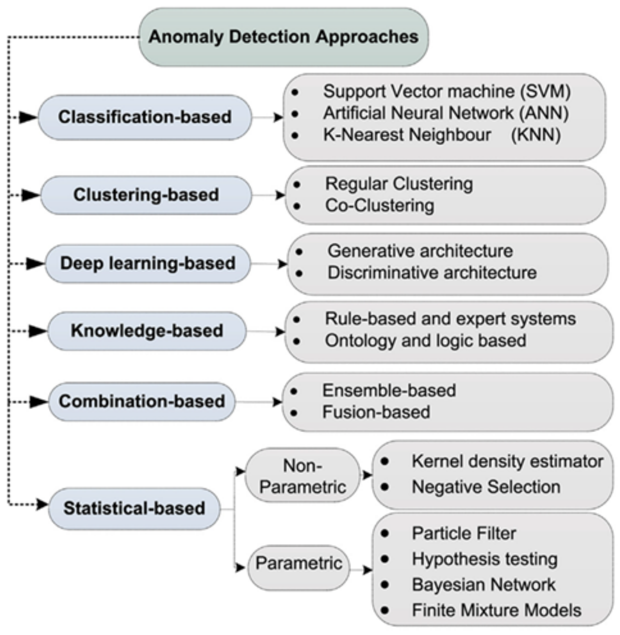

Figure 1: Taxonomy of Anomaly Detection [2]

Detection of misuse and/or intrusion involves identifying potentially suspicious activities within a network or on hosts. Misuse detection focuses on recognizing deviations from established rules by individuals with valid system access rather than actual intrusions. For instance, when an employee uses the Internet for personal purposes in violation of company policy, it constitutes a misuse intrusion. In contrast, intrusion detection is designed to identify unauthorized individuals, such as external hackers or government spies, who lack authorized system access. The intrusion detection system primarily focuses on spotting ongoing intrusions within the system or network but does not proactively prevent malicious activities.

There are two main types of methods for detecting intrusions: signature-based systems, known as SIDS, and anomaly-based systems, referred to as AIDS. Anomaly detection systems are further categorized into network-related and host-related intrusion detection systems. The identification of normal or anomalous data in anomaly detection techniques is achieved through the utilization of labels.

A SIDS identifies suspicious activities through pattern matching with known external attack patterns, which fall into two categories: misuse detection and knowledge-based detection. This anomaly detection model compares the recent signature with a previously stored signature in its database. When a match occurs between these signatures, the IDS signals the presence of malicious activities within the network. Regular updates to the signature database are crucial to effectively detect malicious activities in a network. However, it's important to note that this type of detection system cannot identify zero-day attacks, as these novel attack types may not yet be contained in the signature archive. This anomaly detection model delivers optimal detection outcomes for recognized signatures associated with malicious activities. This type of anomaly detection model is known for its straightforward configuration and comprehensibility. Widely adopted intrusion detection systems include Snort and NetSTAT. In the traditional setup, the SIDS examines network packets and matches them against stored signatures. However, newly introduced attacks not yet included in the signature database can reduce intrusion detection accuracy. To address these limitations, the anomaly-based anomaly detection model offers enhancements and boosts the overall anomaly detection rate. The anomaly-based intrusion detection approach is designed to identify malicious and unreliable network exploitation activities within the corporation.

AIDS effectively addresses the limitations included in SIDS approaches. Likewise, the anomaly-based intrusion model leverages statistical-based, machine learning, and knowledge-based techniques to model the typical behaviors of network traffic. An "anomaly" refers to any behavior that deviates from these established norms, and such traffic anomalies can harm computers and network devices. Anomaly-based detection can occasionally yield false results due to shifts in user behavior. AIDS can generate errors even when legitimate users alter their usual habits. This approach comprises two key stages: the testing and the training stages. In the training stage, the model is trained using normal traffic data to establish a baseline or "normal profile." In the testing stage, previously unseen data is employed to evaluate the model's performance. The primary benefit of this methodology is its ability to detect zero-day attacks.

Three distinct anomaly detection methods include unsupervised, semi-supervised, and supervised anomaly detection methods based on the target class. AIDS addresses the limitations of SIDS by employing knowledge-based methods, machine learning, and statistical-based to model normal behaviors. Figure 1 outlines the various approaches to anomaly detection [2].

Deep learning has the capability to generate improved representations, enhancing the development of effective anomaly detection models. In contrast, conventional machine learning algorithms for network-related abnormality detection are more appropriate for smaller datasets and often rely on performance influenced by the implementation of feature engineering. The model benchmark indicators of conventional anomaly detection models are significantly influenced by the split ratio. While these conventional ML methods are uncomplicated and require minimal resources, they face limitations when dealing with extensive datasets and large feature sets, making them unsuitable for tasks such as machine vision, image translation, natural language processing, and similar applications.

The convolutional neural network is primarily employed for computer vision using image datasets, with the lower layers' neurons responsible for feature reduction. These lower layers typically recognize image corners, boundaries, and intensity called small-scale features. As the information progresses to higher layers, the network integrates these lower-level features to create forms, basic shapes, and partial objects. The final layer of the network amalgamates these lower-level features to generate the output. LSTM operates distinctively from a CNN as it is commonly applied for processing and predicting outcomes based on sequential data. Unlike CNNs, Recurrent Neural Networks (RNNs), including LSTMs, were specifically designed to preserve long-range information within a sequence, preventing the loss of important details in lengthy sequences. The Bidirectional LSTM (BiLSTM) enhances this by introducing an additional LSTM layer that reverses the flow of information, effectively addressing issues such as vanishing gradients.

The deep learning methodology tackles challenges inherent in conventional machine learning, specifically its ability to handle extensive datasets and numerous features. The efficacy of anomaly detection algorithms based on deep learning depends on various factors, including the choice of hidden layers, determination of activation function, neurons, batch size, and epochs during both model training and testing. Strategic decisions regarding these hyperparameters, along with considerations for the ratio of the train to test data and the design of deep neural networks, are essential for improving the precision of network anomaly detection systems.

In addition to fine-tuning hyperparameters, handling imbalanced data is vital, and creating a balanced dataset using various sampling methods contributes to improved anomaly detection. Under-sampling reduces data size, posing challenges for deep learning models. At the same time, over-sampling methods generate duplicate random data, proving more effective for deep learning models to improve the anomaly detection performance in network-based anomaly detection models.

2. Literature Review

The continuous generation of data generates big data and poses challenges for traditional machine learning algorithms, requiring extensive feature engineering efforts to perform adequately. Deep learning significantly enhances detection performance in such scenarios. However, the effectiveness of network anomaly detection varies on numerous factors, with the nature of the dataset (whether balanced or unbalanced), the hyperparameter of the neural network, the amount of model train and test data, and the architecture of the neural network in the deep learning model. These elements collectively play a crucial role in successfully identifying anomalies in the network.

In their study, the researchers utilized the Bidirectional LSTM to alleviate the considerable requirements for feature reduction inherent in conventional machine learning-based anomaly detection approaches [4]. Additionally, they implemented data augmentation in data preprocessing of minor attacks user to root (U2R) and root to local (R2L) to create a well-adjusted NSL-KDD. This methodology resulted in higher anomaly detection accuracy of 90.73% and f1-scores of 89.65%.

In their study, presented an algorithm for network intrusion detection that integrated a deep hierarchical network with hybrid sampling, incorporating SMOTE to create a balanced dataset [5]. They utilized a hybrid approach that combined CNN and Bi-LSTM for anomaly detection accuracy of 83.58% on NSL-KDD and 77.16% on UNSW-NB15.

In their study, presented a method utilizing bidirectional generative adversarial networks (Bi-GAN) on the CIC-DDoS2019 and NSL-KDD datasets [6]. The Bi-GAN model exhibited strong performance, particularly on the imbalanced NSL-KDD, achieving an f1-score of 92.68% and an accuracy of 91.12%. The Bi-GAN approach was employed to enhance the performance of the NSL-KDD imbalance dataset.

In their study, implemented a new method involving auxiliary classifier generative adversarial network (ACGAN) and ACGAN with SVM to tackle data unevenness concerns by leveraging GAN to produce synthetic attack network traffic for intrusion detection systems [7]. These artificially generated attacks were merged with the existing data, resulting in an extended dataset. Research carried out on the RAWDATA, CICIDS2017, UNSW-NB15, and NSL-KDD showed that among the support vector machine, decision tree, and random forest models, the decision tree achieved a superior f1-score of 92% on the balanced NSL-KDD dataset.

In [8], researchers utilized an assorted ensemble-aided approach to binary and multi-class network anomaly detection models to tackle the challenge of uneven traffic data in network traffic-related datasets, including NSL-KDD, UNSW-NB15, and KDD99 datasets. This approach achieved a true positive rate and area under the ROC curve of 94.5% and 96.2% on the NSL-KDD, respectively.

According to their finding, the authors [9] concluded that the efficiency of the anomaly detection algorithm is improved when the number of output labels is reduced. This observation was explored across different conventional machine learning algorithms, including Naïve Bayes, J48, random forest, bayesinNet, bagging, and bayesinNet. The evaluation used three network datasets: KDD99, CICIDS2017_Thrusday, and UNSW-NB15.

In [10], the researchers observed the effectiveness of a recurrent neural network-based intrusion detection system (RNN-IDS) in multi-class and binary-class scenarios. Performance on the NSL-KDD was observed, which is affected by the number of neurons and different learning rates. Experimental outcomes illustrated that RNN-IDS is adept at constructing a classification approach with high accuracy, outperforming traditional machine learning classification methods, including random forest, artificial neural network, J48, and support vector machine in both multiclass and binary network intrusion-related datasets. In their publication [11], presented a network anomaly detection technique utilizing a convolutional autoencoder and attained a model accuracy of 96.87% on the NSL-KDD. The convolutional autoencoder methodology was utilized to simplify and determine the most significant features of the network anomaly dataset.

In [12], the authors investigated the usefulness of several autoencoders in detecting network anomalies. They compared four different types of autoencoders, including sparse autoencoders, undercomplete deep autoencoders, and denoising autoencoders, using the NSL-KDD. Sparse deep denoising autoencoder yielded a model accuracy of 89.34% compared with other models.

In [13], the authors presented a model centered around a 5-layer autoencoder (AE) tailored for network abnormality detection. The fine-tuned model designs demonstrated proficiency in attribute learning and the dimension of data reduction, resulting in improved performance metrics, including model accuracy and f1-score. The model produces the highest accuracy and f1-score of 90.61% and 92.26% on the NSL-KDD, respectively. The researchers employed the reconstruction error to determine whether the network traffic is regular or attacked.

In [14], [15], the researchers proposed a network anomaly detection approach with a combination of convolutional neural networks and bidirectional LSTM applied to the KDD99. They explored the influence of the number of nodes, the number of hidden layers and memory elements on it, and the number of epochs to improve their anomaly detection model accuracy. The performance metrics of different models, such as J48, k-nearest neighbors, NB, deep forest, RF, and convolutional neural network combined with bidirectional LSTM, were evaluated. The convolutional neural network bidirectional LSTM exhibited the ultimate model detection accuracy of 95.40%.

In [16], the researchers assessed both single-layer and four-layer LSTM models for weather forecasting, utilizing a weather-related dataset from Hang Nadim Indonesia Airport. The top model validation accuracy of 80.60%. The four hidden layers comprise 50, 90, 100, and 200 memory elements. The split ratio for testing and training dataset was used at 0.30, and the models underwent training for 500 epochs.

The researchers in [17] adopted a deep learning approach utilizing bidirectional LSTM, implemented on the UNSW-NB15 and KDDCUP99, and achieved notable outcomes with a 99% accuracy rate for both datasets. Numerous current models encounter difficulties in effectively detecting uncommon attack traffic types, notably user-to-root and remote-to-local traffic, which often demonstrate lower detection accuracy than other types of attacks. The researchers in [18] deployed an intrusion detection system based on bidirectional LSTM to address the mentioned encounters on the NSL-KDD. This anomaly detection approach, utilizing Bi-LSTM, achieved a model detection accuracy of 94.26 % for binary NSL-KDD data.

The study in [19] delved into the influence of batch size and learning rates on the performance of CNN, focusing on image classification, particularly in the context of medical images. The results indicate that a larger batch size does not necessarily lead to higher accuracy. Moreover, the choice of learning rate and optimizer significantly affects performance. The authors found that reducing the learning rate and batch size, particularly during fine-tuning, enhances the network's training effectiveness.

Diverse strategies were implemented to address the challenge of data imbalance, encompassing techniques such as data augmentation discussed in [4], application of SMOTE detailed in [5], the use of GAN technology explored in [6], [7], the assistance of Heterogeneous ensemble methods investigated in [8], and the reduction of the target class by combining smaller classes into a new category as discussed in [9]. A considerable number of research endeavors in the realm of deep learning for network anomaly detection have been scrutinized, incorporating methodologies like RNNIDS outlined in [10], CAE featured in [11], Autoencoder examined in [12], multilayer AE explored in [13], convolutional neural network combined with bidirectional LSTM hybrid methods presented in [14] and Bi-LSTM discussed in [17]-[18].

The researchers in [17] and [18] did not provide details on data pre-processing, the train-test split ratio, or adopting bidirectional LSTM hyper-parameters in their model study. Similarly, the researchers in [16] conducted weather forecasting using Bi-LSTM without specifying the hyperparameter values. In [10], there was no analysis information on epochs and the train test split ratio for the KDDTrain+ dataset. Most of the literature reviewed emphasizes enhancing model accuracy in conventional or deep learning algorithms. However, there is a notable lack of focus on deciding on hyper-parameters in deep learning approaches, determining the train test split ratio, and defining the architecture of neural networks. Some researchers do not clarify how these values are applied in their work. Consequently, our research aims to address these limitations in the network anomaly detection approaches by conducting experiments on NSL-KDD.

3. Contributions

The literature review examines a gap in the existing network intrusion detection systems during anomaly detection. The primary contribution of this research is to bridge this gap by proposing network anomaly detection models specifically tailored for imbalanced multiclass datasets.

Arbitrarily selection of hyperparameters does not yield efficient anomaly detection performance on the given dataset. This research investigates the impact of epochs, batch size, and optimizers on the efficacy of a bidirectional LSTM anomaly detection model using the multiclass NSL-KDD.

The choice of the amount of training data and testing data also influences the model's performance. A larger training dataset requires a longer training time, whereas a smaller dataset leads to quicker model training. The model's efficacy is contingent on the data size utilized for both training and testing, a factor we explored by adjusting the test train split ratio to enhance the performance of network traffic anomaly detection on the NSL-KDD.

More layers add complexity to the neural network-based model. The program execution time (program training and testing time) is large compared to small numbers of neural network layers and memory elements. The memory elements and layers in the neural network architecture influence the network anomaly detection performance. Investing layers and memory elements of neural networks improves the bidirectional LSTM on the NSL-KDD.

The careful choice of machine learning and/or deep learning algorithms significantly impacts the effectiveness of network anomaly detection. This study introduces the creation and deployment of a network traffic anomaly detection system utilizing a bidirectional LSTM-based recurrent neural network model. The developed model demonstrates a remarkable anomaly detection accuracy of 98.52 % in the network, particularly for the binary NSL-KDD.

The primary challenge when dealing with real datasets is the presence of imbalanced data. Various approaches can be employed to address this issue. In the NIDS multiclass dataset, both under-sampling and over-sampling methods are applied to tackle data imbalance. Notably, oversampling methods proved to be more effective, achieving the highest detection accuracy of 99.83% for the multiclass NSL-KDD datasets.

4. Model Description

The proposed model consists of different steps, which are listed:

1. Data collection and modelling
2. Data cleaning and pre-processing
3. Bidirectional LSTM model preparation
4. Model training and testing
5. Evaluation model
6. Compare the model for decision

Figure 3 illustrates the schematic for the model based on Bidirectional LSTM. More elaborate explanations of the methods outlined above for the proposed model will be provided in the following sections.

4.1. Data Collection and Modelling

During this study, we employed the KDDTrain+ dataset, one of the subset data from the NSL-KDD.

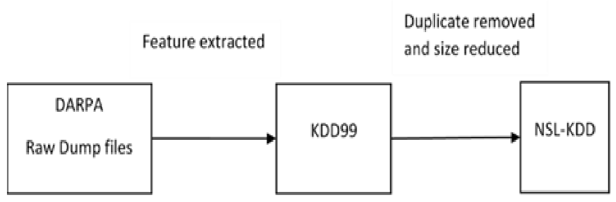

Figure 2: DARPA, KDD99, and NSL-KDD Datasets

The NSL-KDD data is derived from the DARPA KDD99 data, as depicted in Figure 2, after the removal of noise and unwanted data. This includes the complete training data from the NSL-KDD set, including features named attack_type and difficulty. It encompasses 41 attributes and covers five separate attack categories: denial of service, normal, remote_to_local, probe, and user_to_root. NSL-KDD [20] represents an enhanced version of the KDD99 network traffic anomaly data, eliminating duplicate entries in the training data and ensuring the absence of repeated records in the test data. The KDDTrain+ dataset comprises 125,973 records and includes 41 attributes. Notably, this is balanced, with 53.46% of total traffic being normal and 46.54% of total traffic entry being abnormal. We picked this data because it is balanced data between normal and abnormal traffic records within the subset, making it suitable for binary network anomaly detection data. Those numbers of attack class information from the NSL-KDD data were utilized to create the multiclass dataset for the experiment, detailed in the data pre-processing section.

4.2. Data Cleaning and pre-processing

The KDDCup99 data is widely employed in experiments related to anomaly detection in computer network traffic. It consists of network-related traffic that transfers from the virtual network environment utilized for the third knowledge discovery and data mining tools competition. The KDD99 network traffic data is a revision of the 1998 DARPA. The KDDCup99 dataset comprises three components: the "Whole" dataset, the "10% KDD," and the "Corrected KDD." The "Whole" dataset encompasses various attack traffic and one normal network traffic connection. This data involves two training data subsets: a full training data subset and a 10% training data subset. The "Whole" dataset consists of 4,898,431 individual records containing 41 attributes labeled as normal or an attack.

As indicated in reference [20], the KDD99 dataset encompasses 22 distinct attack traffic categorized into four classes: Denial of Service, Unauthorized Access to Local Privileges (U2R), Unauthorized Remote Machine Access (R2L), and Scan Network (Probe). The NSL-KDD data contains four sub-datasets, including KDDTest-21, KDDTest+, KDDTrain+_20Percent, and KDDTrain+. Notably, the KDDTrain+_20Percent and KDDTest-21 portions are sub-datasets derived from the KDDTest+ and KDDTrain+, respectively.

The KDDTrain+ dataset is designated as the training dataset, while the KDDTest+ dataset serves as the testing dataset for the machine learning model. KDDTest-21 is a subset of the test dataset that excludes the most challenging traffic records (with a score of 21), and KDDTrain+_20Percent is a subset of the training dataset, encompassing 20% of the entire training dataset. It's important to note that the traffic records found in KDDTest-21 and KDDTrain+_20Percent are already included in the test and train datasets, respectively. The NSL-KDD dataset addresses the limitations found in the KDD'99 dataset. Unlike KDD'99, NSL-KDD ensures the absence of redundant values in both the train and test datasets.

Notably, NSL-KDD is advantageous due to its smaller test and train sets, eliminating the need for random selection of a small data subset, thus making experiments more cost-effective. Each record in the NSL-KDD dataset comprises 42 features, with 41 of them corresponding to the traffic input and the final label denoted as either "normal" or "abnormal." In the KDDTrain+ contains 125,973 total network traffic records and 41 generated attributes, the data cleaning and pre-processing assigns a target label of ‘1’ for normal traffic and ‘0’ for attack traffic records, transforming the multiclass network traffic data into a binary class.

achine learning and deep learning algorithms work only for numeric values, so ‘'protocol_type,' 'service,' and 'flag' are categorical attributes transformed into numeric values, either ‘0’ or ‘1’ using one hot encoding method called dummy one hot encoding. The dataset is then normalized using the standard scalar method. Correlation-based feature reduction is also implemented where those features with a correlation factor exceeding 0.5 are preserved to reduce the features. Binary class data is employed in experiments A to E. In experiment F, the multiclass (class 5) version of the NSL-KDD dataset is utilized. Prior to training and testing the BI-LSTM model, a sampling method is applied to balance the unbalanced multiclass data. Further details on data preprocessing and model information can be found in the experimental section in the subsequent chapter.

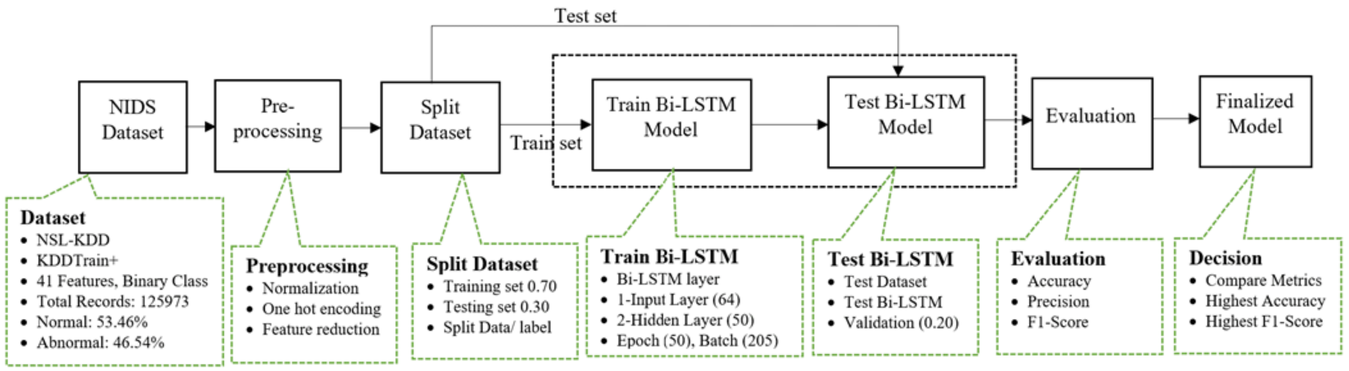

Figure 3: Bidirectional LSTM model block diagram

4.3. Train and test data preparation

The train data and test data splitting method separate the data randomly into two different subsets of the dataset. These two subsets of data contain the designed amount of data based on our selection. Since our pre-processed dataset represents just one portion of the data, we employ two separate datasets for implementing the machine learning algorithms. Researchers typically have flexibility in determining the train-test split ratio, with common choices of 80% to 20%, 60% to 40%, 70% to 30%, and 75% to 25%. We conducted experiments to determine our model's most optimal splitting ratio and found that a 70% training and 30% testing dataset ratio yielded the best performance.

4.4. Bidirectional LSTM model preparation

A recurrent neural network comprises feedback paths that analyze data sequences and patterns to make predictions. These loops enable data sharing among nodes, facilitating predictions based on accumulated information referred to as memory. RNNs have been effectively applied to address machine learning challenges, including tasks such as language preprocessing models, human voice/ speech recognition, and image processing.

The LSTM-based model resolves the challenge of vanishing gradients encountered in RNNs. The LSTM architecture comprises a memory block and three units: input gates, output gates, and forget gates. These gates function similarly to read, write, and reset functions for the cells. Due to the presence of those three gates, LSTM memory cells can effectively store and retrieve data over prolonged times, mitigating the issue of vanishing gradients.

Conventional RNNs are limited in their capacity only to consider past context information. In contrast, Bidirectional RNNs overcome this constraint by analyzing data in forward (left to right) direction and backward (right to left) directions. This involves integrating two hidden layers, with the outcomes subsequently forwarded to a shared output layer. In a conventional LSTM neural network, the output signal/data is generated directly. In contrast, a bidirectional LSTM neural network incorporates both directions (forward and backward) layers at each stage, contributing the signal to the neural network activation layer. This configuration captures data from both preceding and succeeding data, allowing the bidirectional LSTM neural network model to predict the target sequence of each element by considering finite sequences in the circumstances of both past and future elements. This is achieved by employing two consecutive LSTMs—one processing data from both directions. Traditional RNNs are constrained by their dependence solely on the previous perspective. Bidirectional LSTM defeats this limitation by examining data feed from both directions through two hidden neural network layers and then forwarding the results to a similar recurrent neural network output layer.

In a standard LSTM-based model, the model prediction is usually obtained directly via the given dataset. Conversely, the outputs from the forward layers and backward layers from each stage are combined and input into the activation layer in the bidirectional LSTM model. This resulting output encapsulates data from past and future data from the memory blocks in LSTM. The bidirectional LSTM predicts the labels or sequence from each element by leveraging finite sequences within the circumstances of preceding and following items. This process is accomplished through the sequential processing of two LSTMs—one data sequence from right to left direction and the same data sequence from left to right.

The selection of neural network architecture components, including input layers, hidden layers, output layers, layer sizes, activation functions, and dropout rates, is a critical step following data preprocessing. Hyperparameter tuning is an integral part of this research. Initially, hyperparameters are chosen randomly for experimentation, as discussed in more detail in the subsequent experimental sections. The data sampling approaches are implemented to deal with the data unevenness problem. Random oversampling and random under-sampling methods created the balanced multiclass dataset.

To initiate the random selection of the Bidirectional LSTM architecture, the neural network comprises a single input layer with 64 neurons and a dropout rate of 20%. It features two hidden layers with 50 neurons each, both employing a 20% dropout rate. The output layers consist of a single dense layer, and the choice of activation function depends on the nature of the target class size, whether binary or multi-class. Once this model is defined, it is compiled using the appropriate loss function and optimizer in preparation for training.

4.5. Evaluation Bi-LSTM model

Multiple experiments have been conducted to analyze the efficacy of the bidirectional LSTM model, revealing inconsistencies in the effectiveness of both machine learning and deep learning models. Consequently, a comprehensive analysis of the model's hyperparameters becomes imperative for performance improvement. The selection of the optimizer, batch size, epochs, and train test splitting ratio is guided by a comparison of anomaly detection accuracy and f1-score metrics for the bidirectional LSTM model. Ultimately, the bidirectional LSTM's performance metrics are juxtaposed with previous research findings to assess its efficacy. Additionally, two distinct sampling methods, namely random under-sampling and random oversampling, were experimented with on the NSL-KDD and compared using the bidirectional LSTM model.

4.6. Compare performance for decision-making.

After conducting model testing and evaluation, the decision-making process involves selecting the most suitable model pipeline from various alternatives. During this research, multiple sets of experiments are conducted to optimize the hyperparameters for the Bi-LSTM model, aiming to enhance its performance. These hyperparameters encompass factors such as optimizers, epoch count, batch size, neural network architecture, class size selection, and methods for preprocessing raw data. This optimization process is driven by comparing performance metrics obtained from these diverse sets of experiments. Additionally, the performance metrics of the bidirectional LSTM anomaly detection models for NSL-KDD data are compared with published literature results.

5. Experiments and Results

Sets of experiments were conducted on a Windows 10 laptop with a 64-bit architecture, equipped with 16GB of random-access memory and an i7-1.99GHz processing unit. Python3.7.13, Keras2.6.0, and TensorFlow2.9.1 were utilized in this research. The investigation into train and test data split ratio, numbers of epochs, optimizers, and batch size for the bidirectional LSTM model was carried out across various experiments, as elaborated below. The intrusion detection system leverages machine and deep learning techniques for anomaly detection. Python is utilized to code network intrusion detection models, using packages such as NumPy, Pandas, Keras, imblearn, and Sci-kit-learn for developing machine learning models. Additionally, tools like WEKA, Java, C#, Visual C++, and MATLAB are commonly employed in intrusion detection.

To ensure reproducibility, seed values are configured to obtain consistent results across multiple runs on the Jupyter Notebook platform. Subsequently, the experimental results are presented in the form of plots or tables, using the Microsoft Office suite for analysis.

5.1. Experiment: Optimizers Vs. Bi-LSTM performance

During this experimentation, the bidirectional LSTM was applied to the NSL-KDD, the details of which are outlined in the preceding sections. An appropriate optimizer is essential for enhancing the network traffic anomaly detection model's training time and the overall efficacy of the model. The choice of optimizer holds significant importance as it expedites results for the ML/DL model. The choice of the optimization algorithm made by a deep learning practitioner directly impacts both the training speed and the ultimate predictive performance of their model. TensorFlow is an open-source machine-learning library containing nine optimizers: Adam, Ftrl, Adagrad, Adamax, Adadelta, SGD, RMSProp, gradient descent, and Nadam. Among them, seven optimizers were experimented with to achieve the highest performance of the model.

Table 1: Optimizer Vs. Accuracy

training data= 70%, Epochs = 50, batch size= 512					
SN	Optimizer	Accuracy %	Precision %	f1-score %
1	**Nadam**	**98.26**	**97.76**	**98.37**
2	Adam	98.24	97.66	98.35
3	RMSprop	98.19	97.56	98.31
4	Adamax	97.95	97.40	98.08
5	SGD	91.19	88.67	92.02
6	Adagrad	61.86	58.22	73.59
7	Ftrl	53.14	53.14	69.40

In this experimental task, the hyperparameter values were picked randomly, and the performance metrics and optimizers are outlined in Table 1. The structure of the bidirectional LSTM model contained 64 units, featuring two B-LSTM hidden layers having 50 units in each, along with the dense output layer. Each layer within the BLSTM model utilized an activation function called relu and 20% drop-out rate of 20%.

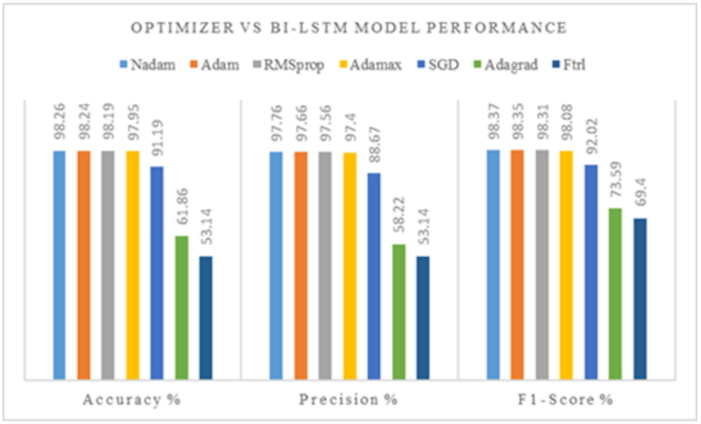

Figure 4: Optimizer Vs. Bi-LSTM performance

Observing the above results (Table 1 and Figure 4), it is determined that the Nadam optimizer is the victorious optimizer, with the winning performance metrics having an accuracy of 98.26%, precision of 97.76%, and f1-score of 98.37%. Nadam enhances the Adam algorithm by integrating Nesterov momentum, resulting in an improved performance of the Adam optimizer.

5.2. Experiment: Train test split ratio Vs. performance

In this experiment, we investigated the impact of both the train test split ratio and model performances. The process of data splitting is crucial in data science, particularly when preparing machine learning models using the available data.

The train test split methodology is utilized to calculate the efficiency of machine learning algorithms in predicting results from data that were unseen during the model training phase. Once the model gets trained, the test dataset is applied, and no fixed percentage split ratio to divide into training and test sets from the given dataset. The splitting ratio is explored to enhance the model performance by utilizing the Nadam optimizer on binary NSL-KDD data.

Table 2: Train test split ratio Vs. performance

optimizer = Nadam, Epochs = 50, Batch_size = 512			
Testing data %	accuracy %	precision %	f1-score %
10	98.15	97.55	98.29
20	98.21	97.57	98.33
30	**98.24**	**97.66**	**98.36**
40	98.18	97.52	98.30
50	98.13	97.50	98.26
60	98.10	97.52	98.24
70	98.12	97.65	98.25
80	97.82	97.39	97.97
90	97.92	97.31	98.07

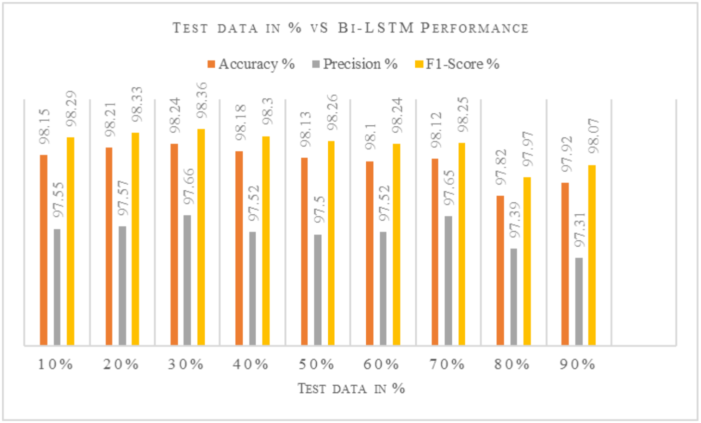

Figure 5: Test data size in % Vs. Bi-LSTM model performance

This experimental work presents the train test split ratio that achieves the optimal performance for our network traffic anomaly detection model on the NSL-KDD. The performances are tabulated in Table 2. and the plot is shown in Figure 5, where a 30% test split percentage results in the model's highest accuracy of 98.48% and f1-score of 98.57%.

5.3. Experiment: Batch size Vs. performance

This experimental work presents the train test split ratio that achieves the optimal performance for our network traffic anomaly detection model on the NSL-KDD. The performances are tabulated in Table 2. and the plot is shown in Figure 5, where a 30% test split percentage results in the model's highest accuracy of 98.48% and f1-score of 98.57%.

Table 3: Batch size Vs. model performance

Optimizer = Nadam, epochs = 105, testing data split= 0.30			
batch size	**f1-score %**	**accuracy %**	**prgm exe time (sec)**
50	**98.58**	**98.48**	2127.235
500	98.47	98.36	514.770
350	98.51	98.4	527.153
450	98.51	98.41	454.989
250	98.46	98.35	616.466
150	98.52	98.42	858.070
300	98.55	98.45	553.444
200	98.55	98.45	796.898
400	98.48	98.38	460.884
15	98.56	98.45	5671.738
100	98.56	98.46	1228.779

A smaller batch size entails the introduction of limited data samples into the Bi-LSTM anomaly detection model, necessitating a lengthier training period than a larger batch. The performance metrics and batch size are presented in Table 3. The experimented results indicate that when applying this model to the NSL-KDD, a batch of 50 produces optimal accuracy and s1-score. A larger batch of data through the model takes less training time but exhibits lower accuracy, highlighting a significant trade-off for this Bi-LSTM network traffic anomaly detection model.

5.4. Experiment: Epochs Vs. performance

In machine learning, an epoch represents one complete pass through all the training data during a model's training. During each epoch, the model is exposed to the entire dataset, and the model's parameters (weights and biases) are adjusted based on the error or loss calculated from the model's predictions compared to the actual target values.

Table 4: Epochs Vs. model performance

optimizer = Nadam, batch= 50, test data= 30% , train data = 70%			
epoch	accuracy	f1-Score	prgm exe time (sec)
175	98.48	98.58	3965.207
100	98.48	98.58	1878.803
125	98.48	98.58	2470.620
5	97.9	98.03	127.058
35	98.35	98.46	761.278
205	**98.52**	**98.62**	4103.767
50	98.38	98.48	942.129
45	98.37	98.47	1002.092
75	98.46	98.56	1465.514
150	98.48	98.58	2934.249
25	98.3	98.41	527.524
15	98.13	98.25	322.529
Accuracy and f1-score in %, prgm exe time:: program train and testing time			

In practice, the epoch is a hyperparameter set before the training begins. The choice of the epoch size depends on factors such as the model's complexity, the data size, and the model's convergence behavior during training. Selection of a small epoch may result in model underfitting, where the machine learning model hasn't learned the underlying patterns in the data. However, a large size epoch may lead the model to overfit, where the model starts memorizing the training data instead of generalizing well to unseen data. The epoch selection can be any integer value that lies between 1 to infinity. By tradition, the ML/ DL researcher selects large values of epochs.

This experiment aims to identify the optimal number of epochs that yield the highest accuracy for the Bi-LSTM model. Similar to the previous experiment, the Bi-LSTM hyperparameters were randomly selected. Longer epochs result in extended training times for the model. The random numbers of epoch values were chosen between 5 to 205, and the accuracy and f1-score were found to be highest at 205 epochs. However, it's important to note that a larger epoch value increases the training time for our model. In this experiment, a batch of 205 sizes enhances the accuracy of the Bi-LSTM network traffic anomaly detection model, achieving a detection rate of network anomalies at 98.5%.

5.5. Experiment: Model layers parameters Vs. accuracy

In our prior experiments, 5.1 to 5.4, we investigated the impact of various hyperparameters, including the optimizer, number of epochs, batch size, and the train test data split ratio. Our results reveal that the combination of the Nadam optimizer, 205 epochs, a batch size of 50, and a train test split ratio of 70%: 30% delivers optimal performance after evaluating the model performance metrics.

Table 5: Bi-LSTM architecture Vs. accuracy

optimizer = Nadam, batch_size = 50, test data= 30%, train data=70%						
Input layer		Hidden layer 1		Hidden layer 2		acc. %
neuron	act. fn	neuron	act. fn	neuron	act. fn	
8	relu	8	relu	8	relu	97.48
4	sigmoid	4	sigmoid	4	sigmoid	97.05
16	relu	16	relu	16	relu	97.93
16	selu	16	selu	16	selu	97.97
64	**sigmoid**	**50**	**sigmoid**	**50**	**sigmoid**	**98.52**
49	sigmoid	128	sigmoid	128	sigmoid	98.18
80	relu	64	relu	64	relu	98.48
4	relu	4	relu	4	relu	97.55
act.fn::activation function, acc:: model accuracy						

This study investigated different configurations of neurons and activation functions for the neural network of the Bi-LSTM model. The dense output layer is structured to provide probabilities for distinguishing between normal and abnormal classes, rendering the softmax activation function the most appropriate selection for the binary class dataset.

This experiment evaluated diverse configurations of Bi-LSTM neurons and activation functions for input and hidden layers. Several results from the conducted experiment are outlined in Table 4. Based on the tabulated results, 64 neurons in the input layer and 50 neurons in each hidden layer of our model produce the ultimate accuracy of 98.52 % in the domain of network anomaly detection.

5.6. Experiment: Sampling Vs. performance metrics for multiclass NSL-KDD dataset

Since these data represent a refined version of the KDD99 dataset, minimal data preprocessing is required. The downloaded train data (KDDTrain+) with the target class was initially separated from the training dataset to establish the class label. Among the remaining numerical features, three categorical attributes, 'protocol_type,' 'service,' and 'flag,' are extracted. Dummy one-hot encoding methods convert categorical into numerical values, while the numerical features are normalized using standard scaling methods. Subsequently, both feature sets are merged into a unified data frame, resulting in the final data set.

The attack types on both KDD99 and NSL-KDD are presented in Table 6. The network attack traffic in these datasets is classified into 'Denial of Service,' 'Probe,' 'Remote to Local,' and 'User to Root' [21]. A denial-of-service attack prevents legitimate users from accessing resources via the network, causing a disruption in the availability of those resources. On the other hand, a probe is a scanning attack aimed at identifying vulnerabilities in a system connected to the network. This probing attack targets weaknesses and facilitates potential compromise of the system.

Table 6: Attack types and traffic information in NSL-KDD

Class	Attack Types	Data
Probe	Satan, MScan, Upsweep, Saint, Nmap, Portsweep	11656
U2R	Ps, Perl, Buffer_overflow, Sqlattack, Rootkit, Loadmodule, Xterm	52
Normal		67343
R2L	Spy, Ftp_write, Guess_Password, Imap, Phf, Multihop, Warezmaster, Xlock, Warezclient, Xsnoop, Snmpguess, Snmpgetattack, Named, Httptunnel, Sendmail	995
DoS	Back, Worm, Apache2 Neptune, Smurf, Pod, Teardrop, Udpstorm, Processtable, Land	45927
Total traffic data		12593

Likewise, the remote-to-local attack involves illegal access to a remote terminal. The user-to-root attack entails gaining privilege as a root user, with the root password obtained through various techniques such as password sniffing, brute-forcing, or social engineering.

Under-sampling is a straightforward approach and a method for addressing the class imbalance in datasets. This technique involves preserving all data within the minority class while reducing the volume of data in the majority class. It represents one of several tools available to data scientists for enhancing the accuracy of insights extracted from initially imbalanced datasets. In under-sampling, data samples from the majority class are randomly chosen and removed until a balanced distribution is achieved. This reduction in data volume can alleviate storage constraints and enhance processing efficiency. However, it's significant to note that this reduction may result in the loss of valuable information.

Conversely, oversampling is employed when the available data is insufficient in quantity. Its objective is to rectify dataset imbalance by augmenting the number of rare samples. Instead of discarding abundant samples, oversampling techniques generate new rare samples through replication, bootstrapping, or SMOTE (Synthetic Minority Over-Sampling Technique). SMOTE, which stands for synthetic minority over-sampling technique, is a specific form of oversampling that involves the synthetic generation of data points for the minority class. In this process, a random selection of k nearest neighbors is chosen to determine the appropriate oversampling level.

After preprocessing, the NSL-KDD KDDTrain+ multiclass data initially exhibits imbalanced class distributions. Various techniques can be employed to rectify this imbalance, including under-sampling, over-sampling, and hybrid sampling. Our experiment utilized an automated sampling approach combining random under-sampling and SMOTE to restructure the data for all classes based on our implemented sampling method. Random oversampling consists of randomly choosing instances from the minority class, replacing them, and incorporating them into the training dataset. On the other hand, random under-sampling entails randomly selecting instances from the majority class and removing them from the dataset.

Table 7: Bi-LSTM with random under-sampling and performance

BI-LSTM Model with Random Under-Sampling and Performance				
Epochs= 50, Batch_size= 512, Data = NSL-KDD Multiclass (5 class) _RUS				
SN	**Class**	**Precision %**	**Recall %**	**F1-Score %**
1	DoS	100	100	100
2	Probe	100	79.17	88.37
3	R2L	88.89	80	84.21
4	U2R	73.68	93.33	82.35

5	Normal	86.67	100	92.86
Average		**91.29**	**89.74**	**89.81**
Accuracy = **89.74** %				
Program exe time = 17.72 sec				

The number of new datasets generated depends on each target class's original data size. Random under-sampling reduced the NSL-KDD data to 52 instances in each of the five classes by randomly eliminating data points. Conversely, SMOTE, an oversampling technique, augmented the dataset by introducing additional data points. During this experiment, substantial data augmentation created well-balanced datasets, with each target class containing 67,343 instances.

The balanced NSL-KDD data has been partitioned into training and testing subsets to facilitate the training and evaluation of the Bidirectional LSTM model. As determined in previous experiments, the train test data split ratio of 70%:30%.

The architecture of the Bi-LSTM neural network mirrors that used in prior experiments, with the input layer containing 64 elements and both hidden layer1 and hidden layer2 comprising 50 elements. A trade-off analysis was conducted to determine the optimal combination of epochs and batch size while considering the Bi-LSTM model's performance.

Table 8: Bi-LSTM with SMOTE technique and performance

BI-LSTM Model with SMOTE and Performance				
Epochs= 50, Batch size= 512, Data = NSL-KDD Multiclass (5 class) _RUS				
SN	**Class**	**Precision %**	**Recall %**	**F1-Score %**
1	DoS	99.99	99.98	99.98
2	Probe	99.99	99.98	99.98
3	R2L	99.99	99.18	99.59
4	U2R	99.18	1	99.59
5	Normal	1	99.99	1
Average		**99.83**	**99.83**	**99.83**
Accuracy = **99.83** %				
Program exe time = 770.52 sec				

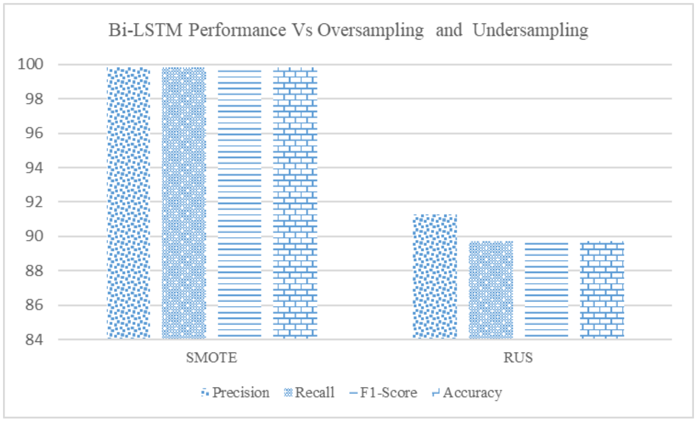

Figure 6: Bi-LSTM performance Vs. oversampling and under-sampling

In our hyperparameter tuning, we aimed to balance program execution time and model performance, as previously demonstrated. As a result, the model was trained for 50 epochs using a batch size of 512 and the Nadam optimizer with a learning rate of 0.041, as detailed in the accompanying table.

The random under-sampling methods produce the NIDS multiclass accuracy of 89.74%, average precision of 91.29 %, recall of 89.74%, and 89.91% f1-score referenced from Table 7. The program execution time is short as compared with oversampling.

Table 8. Shows the performance of the Bi-LSTM with over-sampling methods called SMOTE where the default value of K, i.e., 5, is taken during this experiment. The nearest neighbors value K defines the neighborhood of samples to generate the synthetic samples. We listed the individual class performance as well as average class performance. Figure 6 shows the visualization plot to compare the under-sampling and over-sampling performance on the NSL-KDD multiclass dataset using the Bi-LSTM model. The over-sampling (SMOTE) for the NSL-KDD multiclass dataset provides the 99.83% average precision, recall, and F1 score.

6. Conclusion

The Highest performance is achieved during network traffic anomaly detection using the bidirectional LSTM model. The combination of tunned different hyperparameters (from the above experiments) values, including epoch, optimizer, and batch size, outperformed the anomaly detection model. Determination of hyperparameters' values for the Bi-LSTM anomaly detection model on the NSL-KDD dataset highly contributes to the domain of anomaly detection using machine learning and deep learning. Similarly, we can use no fixed split ratio values for the efficient anomaly detection model. This research work determines the split ratio to produce the highest performance on anomaly detection using the Bi-LSTM model on the NSL-KDD dataset. The combination of neural network architecture memory elements plays an important role in training and testing the model during network anomaly detection. Data imbalance is another main problem to deal with during network anomaly detection. The sampling techniques either delete the data entry randomly or generate the data entry randomly. The sampling technique balances the data in the multiclass dataset. During this research work, the implementation of the random up-sampling methods outperformed the model and produced the highest performance.

We compare our results with existing research [17] to prove that our model is outperformed on the KDD-NSL multiclass dataset. The previously completed research compared their model performance in paper at 99.70% with the other previously researched model's performance, such as Artificial Neural Network (ANN) model at 95%, Decision Tree and Random Forest with 92.60%, Linear Regression, and Random Forest with 94%, Random Forest, and Bayesian Network with 93.4 %, Deep Neural Network with 97% [17]. Our proposed model pipeline for the Bi-LSTM-based network anomaly detection model delivers a higher accuracy of 99.83% is greater than the obtained model performance in research work [17]. The values of bidirectional LSTM model hyperparameters, including epochs values, optimizer, batch size, train test slit ratio, and SMOTE sampling technique for the multilayer bidirectional LSTM neuron architecture (layers, activation function, and memory units) are examined to achieve the highest anomaly detection model performance. The results from these experiments consistently demonstrate that the bidirectional LSTM model, configured with the explored parameters, significantly enhances detection accuracy and f1-score. This model can be experimented with using different network intrusion datasets. Creating a new network intrusion dataset with the latest network attacks will be the extension of this task in the future.

Conflict of Interest

The authors declare no conflict of interest.

Acknowledgment

The National Science Foundation (NSF) and Scholarship for Service CyberCorps (SFS CyberCorps) programs support this research study. The award information of NSF and CyberCorps are #1910868 and #2219611, respectively. This cannot be completed without the continuous support of advisors and the Electrical and Computer Engineering Departments of Prairie View A&M University

References

[1] T. Acharya, A. Annamalai, M.F. Chouikha, "Efficacy of Bidirectional LSTM Model for Network-Based Anomaly Detection," in 13th IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2023, Institute of Electrical and Electronics Engineers Inc.: 336–341, 2023, doi:10.1109/ISCAIE57739.2023.10165336.

[2] N. Moustafa, J. Hu, J. Slay, "A holistic review of Network Anomaly Detection Systems: A comprehensive survey," Journal of Network and Computer Applications, **128**, 33–55, 2019, doi:10.1016/j.jnca.2018.12.006.

[3] S. Samonas, D. Coss, THE CIA STRIKES BACK: REDEFINING CONFIDENTIALITY, INTEGRITY AND AVAILABILITY IN SECURITY.

[4] Y. Fu, Y. Du, Z. Cao, Q. Li, W. Xiang, "A Deep Learning Model for Network Intrusion Detection with Imbalanced Data," Electronics (Switzerland), **11**(6), 2022, doi:10.3390/electronics11060898.

[5] K. Jiang, W. Wang, A. Wang, H. Wu, "Network Intrusion Detection Combined Hybrid Sampling with Deep Hierarchical Network," IEEE Access, **8**, 32464–32476, 2020, doi:10.1109/ACCESS.2020.2973730.

[6] W. Xu, J. Jang-Jaccard, T. Liu, F. Sabrina, J. Kwak, "Improved Bidirectional GAN-Based Approach for Network Intrusion Detection Using One-Class Classifier," Computers, **11**(6), 2022, doi:10.3390/computers11060085.

[7] L. Vu, Q.U. Nguyen, "Handling Imbalanced Data in Intrusion Detection Systems using Generative Adversarial Networks," Journal of Research and Development on Information and Communication Technology, **2020**(1), 1–13, 2020, doi:10.32913/mic-ict-research.v2020.n1.894.

[8] T. Acharya, I. Khatri, A. Annamalai, M.F. Chouikha, "Efficacy of Heterogeneous Ensemble Assisted Machine Learning Model for Binary and Multi-Class Network Intrusion Detection," in 2021 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS 2021 - Proceedings, Institute of Electrical and Electronics Engineers Inc.: 408–413, 2021, doi:10.1109/I2CACIS52118.2021.9495864.

[9] T. Acharya, I. Khatri, A. Annamalai, M.F. Chouikha, "Efficacy of Machine Learning-Based Classifiers for Binary and Multi-Class Network Intrusion Detection," in 2021 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS 2021 - Proceedings, Institute of Electrical and Electronics Engineers Inc.: 402–407, 2021, doi:10.1109/I2CACIS52118.2021.9495877.

[10] C. Yin, Y. Zhu, J. Fei, X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," IEEE Access, **5**, 21954–21961, 2017, doi:10.1109/ACCESS.2017.2762418.

[11] Z. Chen, C.K. Yeo, B.S. Lee, C.T. Lau, "Autoencoder-based network anomaly detection," in Wireless Telecommunications Symposium, IEEE Computer Society: 1–5, 2018, doi:10.1109/WTS.2018.8363930.

[12] M. Ganesh, A. Kumar, V. Pattabiraman, "Autoencoder based network anomaly detection," in Proceedings of 2020 IEEE International Conference on Technology, Engineering, Management for Societal Impact Using Marketing, Entrepreneurship and Talent, TEMSMET 2020, Institute of Electrical and Electronics Engineers Inc., 2020, doi:10.1109/TEMSMET51618.2020.9557464.

[13] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, F. Sabrina, "Improving Performance of Autoencoder-Based Network Anomaly Detection on NSL-KDD Dataset," IEEE Access, **9**, 140136–140146, 2021, doi:10.1109/ACCESS.2021.3116612.

[14] J. Gao, "Network Intrusion Detection Method Combining CNN and BiLSTM in Cloud Computing Environment," Computational Intelligence and Neuroscience, **2022**, 2022, doi:10.1155/2022/7272479.

[15] T. Acharya, A. Annamalai, M.F. Chouikha, "Efficacy of CNN-Bidirectional LSTM Hybrid Model for Network-Based Anomaly Detection," in 13th IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE 2023, Institute of Electrical and Electronics Engineers Inc.: 348–353, 2023, doi:10.1109/ISCAIE57739.2023.10165088.

[16] A.G. Salman, Y. Heryadi, E. Abdurahman, W. Suparta, "Single Layer & Multi-layer Long Short-Term Memory (LSTM) Model with Intermediate Variables for Weather Forecasting," in Procedia Computer Science, Elsevier B.V.: 89–98, 2018, doi:10.1016/j.procs.2018.08.153.

[17] P. TS, P. Shrinivasacharya, "Evaluating neural networks using Bi-Directional LSTM for network IDS (intrusion detection systems) in cyber security," Global Transitions Proceedings, **2**(2), 448–454, 2021, doi:10.1016/j.gltp.2021.08.017.

[18] Y. Imrana, Y. Xiang, L. Ali, Z. Abdul-Rauf, "A bidirectional LSTM deep learning approach for intrusion detection," Expert Systems with Applications, **185**, 2021, doi:10.1016/j.eswa.2021.115524.

[19] I. Kandel, M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," ICT Express, **6**(4), 312–315, 2020, doi:10.1016/j.icte.2020.04.010.

[20] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009, 2009, doi:10.1109/CISDA.2009.5356528.

[21] L. Dhanabal, S.P. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms," International Journal of Advanced Research in Computer and Communication Engineering, **4**, 2015, doi:10.17148/IJARCCE.2015.4696.