8-2024

Enhancement Of Network Anomaly Detection Using Artificial Intelligence Techniques

Toya Acharya

ENHANCEMENT OF NETWORK ANOMALY DETECTION USING ARTIFICIAL INTELLIGENCE TECHNIQUES

A Dissertation

by

TOYA ACHARYA

Submitted to the Office of Graduate Studies
of Prairie View A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2024

Major Subject: Electrical Engineering

ENHANCEMENT OF NETWORK ANOMALY DETECTION USING ARTIFICIAL INTELLIGENCE TECHNIQUES

A Dissertation

by

TOYA ACHARYA

Submitted to the Office of Graduate Studies
Prairie View A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved as to style and content by

Annamalai Annamalai Chair of Committee	Mohamed F. Chouikha Co-Chair of Committee
Xiangfang Li Member	Xishuang Dong Member
Ahmed A. Ahmed Member	Annamalai Annamalai Head of Department
Pamela H. Obiomon Dean of Engineering	Tyrone Tanner Dean of Graduate Studies

August 2024

Major Subject: Electrical Engineering

COPYRIGHT PAGE

ENHANCEMENT OF NETWORK ANOMALY DETECTION USING ARTIFICIAL INTELLIGENCE TECHNIQUES

Toya Acharya

ABSTRACT

ENHANCEMENT OF NETWORK ANOMALY DETECTION USING ARTIFICIAL INTELLIGENCE TECHNIQUES

(August 2024)

Toya Acharya, BE, Pokhara University

Chair of Advisory Committee: Dr. Annamalai Annamalai

Co-Chair of Advisory Committee: Dr. Mohamed F. Chouikha

Traditional signature-based network intrusion detection systems, which capture network attributes, are inadequate against zero-day attack. The smaller number of attacks creates an imbalanced dataset, the major problem during anomaly detection. Machine Learning (ML) and Deep Learning (DL) approaches are promising for network anomaly detection because they can efficiently analyze big network traffic data for malicious activities and detect zero-day attacks. The appropriate selection of the ML/DL algorithm, hyperparameter tuning, and techniques, such as sampling methods, ensemble methods, and reduction of number of classes, can enhance the anomaly detection performance of the anomaly detection methods on an imbalanced network intrusion-based dataset.

The efficacy of various traditional ML models such as Random Forest (RF), J48, Naïve Bayes, Bayesian Network, Bagging, AdaBoost, and Support Vector Machine (SVM) is examined. Different combinations of deep learning models, including convolutional neural networks, bidirectional long-short term memory (LSTM) models, ensemble techniques, sampling techniques, and class reduction approaches, are applied to different sets of network-based intrusion datasets (KDD99, UNSW-NB15, CIC-IDS2017). These

experiments are conducted using different tools (WEKA, Jupyter Notebook) on the Anaconda platform.

Investigation results reveal that traditional ML models are suitable for smaller data and low computational power. Deep learning models outperform huge datasets with large numbers of features but require significantly more computational power. The proposed heterogeneous ensemble method, which combines a number of different models along with a wise selection of hyperparameters and class size reduction techniques, has been demonstrated to significantly enhance anomaly detection performance on communication network-based intrusion datasets.

Implementing different sampling techniques on different training and testing dataset combinations provided insight into application sampling techniques to deal with imbalance network intrusion datasets. The sampling is only efficient for the single set of working data, but the class reduction method to deal with class imbalance problems results in more efficient performance in regard to the single or different set of training and testing data given for network anomaly detection. The overall combination of results and conclusions will provide a comprehensive study of artificial intelligence techniques to enhance network anomaly detection in communication networks.

Index Terms—ADASYN, Bi-LSTM, CIC-IDS2017, class reductions, CNN-BLSTM, deep learning, heterogeneous ensemble learning, imbalance dataset, KDD99, LSTM, machine learning, network intrusion detection system, NSL-KDD, Random Over Sampling (ROS), Random Under Sampling (RUS), SMOTE, SMOTEENN, UNSW-NB15.

DEDICATION

This work is dedicated to my parents and grandparents, whose unwavering support and encouragement have been the cornerstone of my academic journey. Their sacrifices, guidance, and love have shaped me into the person I am today, and I am forever grateful for their endless belief in me. To my parents and grandparents, thank you for believing in me and for instilling in me the courage to dream big and the determination to make those dreams a reality. This work is a testament to your enduring influence on my life, and I dedicate it to you with all my love and gratitude.

ACKNOWLEDGMENTS

I extend my deepest gratitude to my advisor, Dr. Annamalai, and co-advisor, Dr. Chouikha, whose expertise, guidance, and unwavering support have been instrumental in the completion of this thesis. Your mentorship has enriched my academic experience and profoundly shaped my intellectual growth. I am truly fortunate to have had the opportunity to work under your supervision.

I would also like to thank my thesis committee members, Dr. Li, Dr. Dong, and Dr. Ahmed, for their valuable feedback, insightful suggestions, and constructive criticism throughout this journey. Your expertise and dedication have been invaluable in refining this work and elevating its quality.

I am endlessly grateful to my family for your unconditional love, encouragement, and understanding. Thank you for always being my pillars of strength. I am also indebted to my friends for their encouragement, companionship, and endless words of encouragement. Your presence has made this journey more enjoyable and meaningful, and I am grateful for the memories we have shared along the way.

Lastly, I would like to express my appreciation to all those who have contributed to this thesis in ways big and small. This thesis is the culmination of the collective efforts of many individuals, and I am humbled by the generosity and kindness I have received throughout this process. Thank you all for being a part of this journey.

TABLE OF CONTENTS

Page

NOMENCLATURE

ADASYN	Adaptive Synthetic Sampling
AIDS	Anomaly-based Intrusion Detection System
AUC	Area Under ROC
Bi-LSTM	Bidirectional Long Short-Term Memory
BLSTM	Bidirectional Long Short-Term Memory
CIA	Confidentiality, Integrity, and Availability
CNN	Convolution Neural Network
DL	Deep Learning
DoS	Denial of Service
IDS	Intrusion Detection System
IT	Information Technology
K-NN	K-nearest Neighbors
LSTM	Long Short-Term Memory
ML	Machine Learning
MVE	Maximum Voting Ensemble
NIDS	Network Intrusion Detection System
OT	Operational Technology
Probe	Probing
PSO	Particle Swarm Optimization
R2L	Root to Local
RF	Random Forest

ROC	Receiver Operating Characteristic
ROS	Random Over-sampling
RUS	Random Under-sampling
SIDS	Signature-based Intrusion Detection System
SMOTE	Synthetic Minority Oversampling Technique
SMOTEENN	SMOTE combined with Edited Nearest Neighbors
SVM	Support Vector Machine
U2R	User to Root

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

INTRODUCTION

I. Overview

A system is deemed secure when the three core principles of computer security—Confidentiality, Integrity, and Availability (CIA)—are effectively upheld [1]. These principles ensure that data is accessible only to authorized users, remains accurate and unaltered, and is available when needed. An Intrusion Detection System (IDS) plays a crucial role in maintaining these security standards by continuously monitoring and analyzing system activities. It detects potential threats by assessing instances where the CIA principles may be violated, thus helping to identify and mitigate risks before they can compromise the system's security.

With the invention of information and technology, the most crucial information is transmitted in the form of bits from source to destination. The transmitted information can be voice, image, or data, containing banking information, personal information, or network traffic. Various tools or methods are available to detect and prevent intruders. Anomaly is a pattern in the dataset that does not fit into the usual behavior of the data, and some detection techniques are required to detect it. Outliers and anomalies are sometimes used interchangeably in the field of anomaly detection. Anomaly detection has numerous applications, including business, network intrusion detection, health monitoring systems, credit card fraud detection, and fault detection in critical information systems. Anomaly detection is important in cyber security for achieving solid protection against cyber adversaries.

This Dissertation follows the style of the Publication Manual of the IEEE standards.

II. Motivation for Research

Information Technology (IT) and Operational Technology (OT) systems are interconnected via a network to communicate properly. The network between those communication devices is the backbone of modern society to establish seamless communication across the globe. Network anomalies, behavior activities deviating from normal behavior, can be indicative of malicious activities such as intrusion attempts, malware propagation, or even system failures. Detecting and mitigating these anomalies is of utmost importance to ensure the integrity, confidentiality, and availability of network resources.

The motivation for research in network anomaly detection using machine learning, deep learning, and artificial intelligence arises from developing more robust, accurate, and efficient solutions to safeguard network infrastructures. By exploring and advancing these technologies, we can achieve the following benefits:

a. Improved Detection Accuracy: machine learning, deep learning, and artificial intelligence algorithms have the potential to identify complex and subtle anomalies that may be overlooked by traditional methods.
b. Early Threat Identification: network attacks are becoming increasingly sophisticated, making it crucial to detect anomalies at their earliest stages.
c. Adaptability to Dynamic Environments: networks are dynamic systems with ever-changing traffic patterns, making it challenging to define fixed rules for anomaly detection. AI-based network anomaly detection can adapt and learn from network traffic patterns, allowing them to evolve and respond to new and emerging threats continuously.

d. Reduced False Positives: false positives can lead to alert fatigue and unnecessary overhead on network administrators. By employing advanced learning algorithms, hyperparameter tuning, and data preprocessing techniques, researchers can develop models that minimize false positives, ensuring that only genuine network anomalies are flagged for investigation.

III. Summary of State-of-the-Art of Network Anomaly Detection

Network Intrusion detections are classified into two categories, these are signature-based IDS and anomaly-based IDS [1]. The main drawback of the SIDS is that it fails to detect zero-day attacks as those new types of attack signatures are not included in the SIDS signature database. AIDS overcomes the SIDS's drawbacks by modeling normal behaviors using machine learning, statistical-based, or knowledge-based methods [1]. Anomaly-based detection can also produce false results caused by changes in user habits. The machine learning model for network intrusion detection is an example of anomaly-based IDS. Anomaly-based IDS efficiently identifies network-based intrusion, including zero-day attacks.

Most classical research either applies a single data set such as NSL-KDD or CUP99 to train their model. Unfortunately, training a model with one specific data does not guarantee its robustness towards certain unknown or modified attack patterns. To achieve it, training a machine learning model with different data can be vital. On the other hand, most of the existing machine learning methods apply a standalone classifier with classical learning and classification ability. On the contrary, research reveals that different algorithms perform differently for the same input data and show varied results. On the other hand, anomalies, being a small entity over large training samples, undergo a class-

imbalance problem and hence push a machine learning model to show higher inaccuracy or false-positive rate. Moreover, unlike the classical standalone classifier-based approach, ensemble learning with the maximum voting concept can be of great significance to ensure the reliability of the proposed NIDS model. The data preprocessing techniques are also used to cope with the data imbalance problems, including class size reduction and sampling. The class sampling techniques manipulate random data depending on the sampling techniques used. Working on reduction class by aggregating minor classes into a new class creates a more balanced dataset to detect anomalies in the computer network.

Deep learning can extract better representations for creating efficient anomaly detection models. The traditional machine learning-based network anomaly detection algorithms are more suited for small datasets and are mostly performance-dependent on how the feature engineering is implemented. The split ratio is one of the dominant elements influencing the performance of traditional machine learning-based anomaly detection methods. The traditional ML methods are simple and have low resource consumption. Still, for huge datasets and large features, poorly performed and traditional ML cannot be worked on computer vision, natural language processing, and image translations.

The deep learning method overcomes the problems in traditional ML, such as being suited for huge datasets and large numbers of features. The performance of the deep learning-based anomaly detection algorithm depends on the number of neurons, number of hidden layers, types of activation function, number of samples (batch size), and epochs (iterations) during DL model training and testing. Selecting those hyperparameters, training testing data ratio, and architecture of neural network in deep neural networks is vital in increasing the detection accuracy of network anomaly detection systems.

CNN is mostly used in image datasets where the lower layer's neurons reduce the network's features, usually identifying important small-scale features, such as boundaries, corners, and intensity differences. Then in higher layers, the network combines the lower-level features to form more complex features such as simple shapes, forms, and partial objects. On the final layer, the network combines the lower features to produce the output or classification results. An LSTM works differently than a CNN because an LSTM is designed to retain long-range information so that the information is remembered and not lost in a long sequence. Bi-LSTM adds one more LSTM layer, reversing the information flow direction and overcoming the vanishing gradient problems.

Machine learning models have been successfully employed in a variety of artificial intelligence tasks, including natural language processing, machine translation, voice recognition, and image classification. However, these models are susceptible to various threats, such as unknown attacks, anomalies, and adversarial examples. To address this, machine learning models are utilized to detect unknown attacks and anomalies. On the other hand, adversarial examples are intentionally crafted to deceive well-trained machine-learning models. These adversarial examples are virtually indistinguishable from the original inputs to human observers, yet they result in misclassification by machine learning or deep learning classifiers. Similarly, the Natural language processing model is also useful for processing the network traffic utilizing the word embedding methods.

The existence of adversarial attacks raises significant concerns regarding the safety of critical infrastructure systems like flight control systems, healthcare systems, self-driving cars, and more. The potential impact of these attacks on such mission-critical

systems cannot be understated. There are a few network intrusion datasets available for research purposes, such as KDD99, NSL-KDD, UNSW-NB15, Kyoto+, and CAIDA.

IV. Problem Statement and Research Objectives

This dissertation research study solves the different problems that arise in anomaly detection using artificial intelligence methods on network anomaly detection datasets, including KDD99, NSL-KDD, and UNSW-NB15. The machine learning platform effectively detects zero-day attacks. The main problem in the network intrusion detection system is the resource and time consumption during the detection of network attacks. So, the machine learning model performance deteriorates the capacity of the intrusion detection system in the highly growing network system. Feature reduction techniques do not always improve the performance of the machine learning model.

The single standalone machine learning models cannot detect the network anomaly efficiently because the different ML models produce different performances based on the nature of the dataset. Similarly, the amount of anomaly data is always lower in ratio to the benign dataset. The class imbalance problems arise during anomaly detection in case of rare attacks. With the increase in the terminals connected to the internet, the amount of data is humongous; hence, the traditional machine learning model cannot handle it. Those deep learning models are not enough to capture past information for future states. Also, those adversarial attacks are created to fool the machine learning and deep learning models during anomaly detection. The hyperparameter selection was not implemented during most of the previous research.

The problem itself is separated into the following sub-problems:

a. The traditional anomaly detection methods have a high false positive ratio.

b. Many attacks class creates an imbalance in the dataset.
c. The selection of a machine learning model for the different network anomaly datasets is challenging. Even anomaly dataset of the same nature acts differently during network anomaly detection.
d. Single algorithms do not always have a high anomaly detection rate on the given network dataset.
e. The time consumption during the training and testing period affects the performance of the intrusion detection model.
f. Selecting instances in the class, training, and testing data size is very challenging, and class imbalance is one of the main problems in the machine learning-based intrusion detection system.
g. The shallow neural network model does not store the past attack information.
h. The duplication of the previous network anomaly detection research work is not easy without hyperparameters information.
i. Adversarial traffic is the latest attack method used by hackers.
j. Label generation during the network anomaly detection dataset is time-consuming and requires a skilled person.
k. Class imbalance problems can be working with the distribution of data in different classes.
l. The crucial selection of hyperparameters for the neural network improves the performance.

The following research questions are considered to solve the above sub-problems.

a. What are machine learning models highly suitable for the given anomaly dataset?

b. Do multiclass or binary class datasets perform better on a network anomaly dataset?
c. What is the effect of target classes on the performance of the machine learning model?
d. Is there any relation between the size of the dataset during training and testing on the Machine learning model?
e. Does only feature reduction increase the performance of the machine learning model?
f. Are those created features highly related to the target class on the intrusion dataset?
g. Do the Recurrent Neural Network-based models enhance the anomaly detection performance?
h. How did those hyperparameters tunning enhance the network anomaly detection performance?
i. How do the different sampling techniques work for the different combinations of training and test datasets?
j. Does class reduction by aggregating the minor classes into a new single class enhance anomaly detection?

V. Anticipated Broader Impact of the Study

The anticipated broader impacts of the study are as follows:

a. Investigate network anomaly datasets, features, and types of network attacks.
b. Select the network anomaly datasets used for measuring the performance of machine learning and deep learning.
c. Investigate the size of train and test data sets during training and testing the machine learning and deep learning.

d. Investigating the performance of machine learning, deep learning, and generative adversarial algorithms on binary and multiclass network anomaly datasets.
e. Determine if the investigated feature correlates to the target class and increases the performance of machine learning and deep learning.
f. Investigate the effect of data distribution on attack classes and the effect of performance of machine learning and deep learning.
g. Investigate the rare attacks on network anomaly datasets and the performance of machine learning and deep learning.
h. Investigate the ensemble methods to enhance anomaly detection performance.
i. Investigate the hyperparameters for deep learning and sequential recurrent neural networks.
j. Perform an experimental study on publicly available network anomaly datasets to investigate the objective of this study.
k. Investigate the different types of sampling methods to deal with the data imbalance problem.
l. Investigate the different combinations of sampling methods in training and testing datasets.
m. Evaluate the experimental results.

VI. Significance of the Study

The significance of this study contributes to the fields of network anomaly detection systems and different artificial intelligence methods. The primary significances and contributions are as follows:

a. To show the relation between the number of the target class and the performance of the machine learning and deep learning model. The reduction of the number of classes increases the performance of anomaly detection models.
b. To ascertain if feature reduction does not always improve the performance of the machine learning model. So, creating highly correlated features with the output class improved the model's performance for anomaly detection.
c. To determine if the performance evaluation of the anomaly detection model varies depending on the training and test methods used during model design. Also, the size of training and testing data produces different performances.
d. To study the different machine learning and deep learning models on different network anomaly datasets to provide insight into creating the required anomaly detection dataset.
e. To enhance the anomaly detection performance using ensemble methods rather than using single ML/DL models.
f. To preserve the past anomaly during the anomaly detection for deep learning-based network anomaly detection models.
g. To study the class reduction techniques to deal with the class imbalance problem for anomaly detection on network datasets.
h. To study sampling and their combinations on training and testing datasets during network anomaly detection using artificial intelligence methods.

VII. Dissertation Outline

In this dissertation research study, the researcher presented different techniques to enhance anomaly detection using machine learning, deep learning, and artificial

intelligence methods. This study investigated how machine learning, deep learning, and artificial intelligence models efficiently detect anomalies on a binary class network dataset as compared to multiclass network datasets. The imbalance of the dataset on target classes degrades the performance of the anomaly detection model. Similarly, the hyperparameters tunning, using the recurrent neural network-based model and adversarial learning, enhanced the performance of the network anomaly detection models.

Chapter 1 provided a detailed study of the heterogenous ensemble techniques to detect the anomaly. The comparison of the different class performances of the different models, including Naïve Bayes, J48, Random Forest, and more, is investigated in Chapter 2. The efficacy of the deep learning models is explained in Chapter 3 and Chapter 4 with hyperparameter tuning for anomaly detection. The deep learning-based anomaly detection is enhanced using the sampling techniques which is explained in Chapter 5 and Chapter 6. Detailed research includes the different sampling techniques along with combinations of training and testing datasets in Chapter 7. The conclusion and future work conclude this dissertation work.

.

CHAPTER 1. EFFICACY OF HETEROGENEOUS ENSEMBLE ASSISTED MACHINE LEARNING MODEL FOR BINARY AND MULTI-CLASS NETWORK INTRUSION DETECTION

Abstract- The exponential rise in internet technologies and allied applications encompassing a significantly large number of networked devices has alarmed academics to achieve more effective and robust security solutions. Undeniably, digitization has led to revolution globally; however, the security threats, breaches, and subsequent losses indicate the need for a robust cybersecurity solution. Unlike classical intrusion detection systems (IDS), network IDS (NIDS) has been becoming more challenging due to continuous changes in attack patterns and anomaly behavior. Solution data-driven machine learning methods have exhibited better by learning over network traffic information and detecting anomalies; however, their generalization over a network with both known and unknown patterns remains questionable. Moreover, most of the classical approaches fail to address the key issues of class imbalance, level-of-significance centric feature selection, normalization, and over-fitting problems, resulting in different performances by varied machine learning models.

In this research work, the novel and robust heterogeneous ensemble machine learning model is developed to detect anomalies in NIDS. The proposed model first applies sub-sampling to alleviate the class-imbalance problem of NIDS datasets. Subsequently, performing normalization using the Min-Max algorithm, it mapped the input data in the range of 0 to 1, thus alleviating overfitting and convergence. The feature reduction is used to reduce the features; it retained the most suitable features without imposing computational overheads, often in meta-heuristic-based approaches. Finally, the proposed

NIDS solution designed a Heterogeneous ensemble learning model with J48, k-NN, SVM, Bagging, AdaBoost, and RF algorithms as base-classifier to perform two-class as well as multi-class classification over feature-selected NSL-KDD, KDD99, and UNSW-NB-15 datasets. Performance assessment in terms of true-positive rate, false-positive rate, and AUC revealed that the proposed NIDS model exhibited better performance than the standalone classifiers and was superior to other existing anomaly detection methods.

Keywords—Heterogeneous ensemble learning, imbalance dataset, network intrusion detection system, machine learning,

I. INTRODUCTION

The majority of the classical research either applies a single data set such as NSL-KDD or CUP99 to train their model. Unfortunately, training a model with one specific data does not guarantee its robustness towards certain unknown or modified attack patterns. To achieve it, training a machine learning model with different data can be vital.

On the other hand, most of the existing machine learning methods apply a standalone classifier with classical learning and classification ability. On the contrary, research reveals that different algorithms perform differently for the same input data and show varied results. It questions the acceptability of standalone solutions for anomaly detection in a real-time cyber-ecosystem. On the other hand, anomalies, being a small entity over large training samples, undergo class-imbalance problems and hence push a machine learning model to show higher inaccuracy or false-positive rate. It demands a data-driven model to employ a specific sub-sampling concept followed by significant feature selection to ensure reliable classification without undergoing local minima and convergence issues. Moreover, unlike

the classical standalone classifier-based approach, ensemble learning with the maximum voting concept can be of great significance to ensure the reliability of the proposed NIDS model.

Considering the above research demands, gaps, and allied scopes, a state-of-art new and robust heterogeneous ensemble machine learning-assisted NIDS solution is proposed for anomaly detection in network flow data in this paper. The heterogeneous ensemble model consists of a number of different base machine learning classifiers for the network-based intrusion detection system. Unlike classical methods, the proposed model embodies multi-level enhancement, including data sub-sampling, normalization, feature reduction, and Heterogeneous ensemble learning to perform intrusion detection over the different NIDS datasets NSL-KDD, KDD99, and UNSW-NB15. Heterogeneous ensemble classifier comprises state-of-the-art, better-performing algorithms like Naïve Bayes, J48 algorithm, k-NN algorithm, SVM, Boosting, Bagging, AdaBoost, and RF were used to perform binary as well as multi-class classification. The simulation performance over the different input network flow datasets revealed that the proposed Heterogeneous ensemble-based machine learning model outperforms other standalone algorithms in true-positive and area-under ROC (AUC).

The remaining sections are divided as follows. Section II discusses the key related works about the IDS and NIDS solutions, followed by the system model in Section III. Simulation results are discussed in Section VI, which is followed by a conclusion and inferences in Section V. References are given in the reference section of this dissertation.

II. RELATED WORK

Authors [2] applied a Support Vector Machine (SVM) algorithm over the NSL-KDD dataset, where the highest detection accuracy was observed as 99.92%. Later, realizing the efficacy of SVM with reduced feature learning, authors [3]applied SVM with KPCA and GA; however, the accuracy of 96% contradicts [2]. It questions the reliability of the performance is projected. In [4], the authors applied particle swarm optimization (PSO) assisted SVM classifier, where PSO was applied as a feature selection model, while SVM performed two-class classification. Interestingly, unlike [2] and [3], authors in [4] could achieve the highest accuracy of 92.90% over the KDD99 dataset. In [5], decision-tree and SVM algorithms were applied for intrusion detection over the KDD99 dataset, where the SVM classifier obtained the highest accuracy of 89.02%. Random forest (RF) algorithm was applied in [6] to perform intrusion detection on the NSL-KDD dataset the detail about ensemble learning is provided on this [7] research paper. The highest accuracy could be obtained by 99.67%, which was higher than the J48-based algorithm. Similarly, in [8], RF algorithm with weighted K-Means clustering applied over the KDD99 dataset to perform intrusion detection. The highest efficiency could be observed as 98.3%. realization limitations of the classical SVM, especially with large data and high-feature size, authors [9] proposed an optimal allocation-based least square SVM (OA-LS-SVM) algorithm for intrusion detection. In [10], the authors applied the Genetic Algorithm (GA) for feature selection and weight estimation of SVM to perform anomaly detection.

Authors [11] proposed a conditional variational autoencoder (CVAE) along with long short-term memory recurrent neural networks (LSTMRNNs) to perform intrusion classification. The authors [12] stated that their proposed model could be superior to other

machine learning models like SVM, k-NN, and Bayesian classifiers to perform anomaly detection and classification. In [13], SVM and Naïve Bayes algorithms were used to perform intrusion classification, where the first was found superior in terms of detection accuracy. Similarly, in [14], Naïve Bayes, SVM, and RF algorithms were used to perform DoS detection in the wireless network. Linear SVM-based lightweight IDS was developed in [15]; however, its superiority over radial basis function (RBF) and polynomial kernel-based SVM seems limited. In [16] applied ANN Bayesian Net-GR algorithm with ANN and Bayesian Net algorithms as a base classifier to perform two-class classification over Gain Ratio (GR) selected features to perform intrusion detection.

A similar effort was made in [17], where a mutual information-based algorithm was designed to perform feature selection followed by two-class classification towards intrusion detection. Realizing that the selection of a suitable set of features can improve accuracy, authors [18] applied the self-taught learning (STL) based IDS model, where deep learning was applied as a feature selection method while SVM performed two-class classification. Principal component analysis (PCA) and linear discriminant analysis (LDA) were applied in [19] with Ant Lion optimization for feature selection, followed by Artificial Neural Network (ANN) based classification to perform DDoS attack detection. In [20], authors used k-NN for data clustering followed by extreme learning machine (ELM) based two-class classification for intrusion detection. Authors [21] designed a transfer learning-based anomaly detection model to explore the common latent information and changes in behaviors. An ensemble model comprising a decision tree, RF, k-NN, and deep NN was designed in [22] to perform intrusion detection using the NSL-KDD dataset. Authors [22] found that the base classifiers DT performed better than other methods, with the highest

accuracy of 84.2%, while the ensemble model exhibited an accuracy of 85.2%. Though the effort made in [23] tried to address the class imbalance problem using SMOTE based sub-sampling followed by RF and AdaBoost based classification; however, AdaBoost was found inferior to RF-based binary classification.

Similarly, in [24], SMOTE was applied with k-NN to perform outlier detection in the NSL-KDD dataset. k-NN with 10-fold cross-validation exhibited satisfactorily. Authors [25] found that SMOTE with RF performs better than the Naïve Bayes classifier for intrusion detection. Authors [26] applied DT and SVM algorithms to perform anomaly detection. Authors found that DT outperforms SVM (99.62%) in a higher true positive rate (99.86%). In [27], the authors applied the Xgboost algorithm to perform intrusion detection in the NSL-KDD dataset. Authors found that Xgboost outperforms SVM, NB, and RF-based IDSs. The authors [28] and [29] used deep learning based NIDS model which are more resource consuming and complex network intrusion detection system.

III. SYSTEM MODEL

The overall proposed model encompasses the following steps.

Step-1 Data Collection and Modelling

Step-2 Data Pre-processing and Normalization

Step-3 Sub-sampling and Consolidation

Step-4 Feature Selection

Step-5 Classification and anomaly detection.

The overall implementation schematic of the proposed NIDS model is given in Fig. 1-1. A detailed discussion of the above-stated methods is given in the subsequent sections.

A. Data Collection and Modelling

In this research, we considered three different datasets, including NSL-KDD, KDD99, and UNSW-NB15. The snippet of the data under study is given as follows:

1) ***KDDCup99.*** KDDCup99 dataset, which is often called the KDD99 data [30], DARPA intrusion detection challenge dataset. This data is available in four key formats, in a full dataset volume and 10% data volume. It has a total of 41 features with five distinct attack classes, Normal, DoS, Probe, R2L, and U2R. Typically, these features are classified into varied groups, such as basic features, content features, and time-based features.

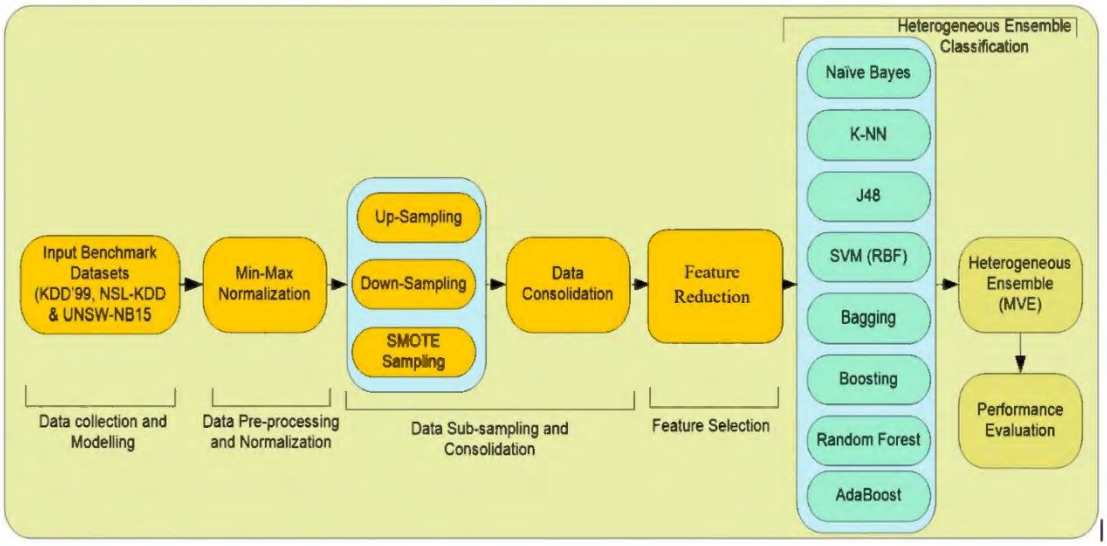

Fig.1.1. Proposed heterogeneous ensemble-assisted NIDS model.

2) ***NSL-KDD Network Flow Data.*** It is a distilled version of the KDD99 intrusion dataset. This specific network intrusion data was created in 2009, and its prime motive was to solve the issues pertaining to non-linear traffic or irregular data patterns in the previous KDD99 dataset. To model NSL-KDD data [31] , the filters are applied to eliminate redundant connection and traffic records in the KDD99 dataset. It possesses a total connection record of 1,36,489. Four sub-datasets are comprised of KDDTest+, KDDTest-21, KDDTrain+, KDDTrain+_20Percent. The portion of NSL-KDD data sets: KDDTest-21 and KDDTrain+_20%, are subsets of the KDDTrain+ and KDDTest+, respectively.

3) ***UNSW-NB15.*** UNSW-NB15 dataset [32] was created by the cybersecurity research team of the Australian Centre for Cyber Security (ACCS), which was specifically created to solve the KDD99 and NSL-KDD dataset's prime issues discussed above. The total data encompassed almost 2 million connection record, However, it is also available in a partial dataset, with 82,332 train connection records along with 175 341 test connection records comprising 10 attacks patterns. The data used in this research comprises a total of 42 features, where a total of nine attacks patterns were used for assessment.

B. Data Pre-processing and Normalization

The input datasets were at first converted into Attribute-Relation File Format (*.arff) for WEKA. Data heterogeneity, imbalanced data nature, skewness, and high data-values variations can force machine learning algorithms to undergo a convergence problem.

C. Data Sub-Sampling and Consolidation

The data sub-sampling and consolidation consists of three different types of sampling methods, up-sampling, down-sampling, and SMOTE. In up-sampling, the arbitrary duplication of the observations or patterns from the minority classes reinforces its signal or value. On the other hand, down-sampling was performed so that it removes observations randomly from the majority class to avoid its presence from dominating the learning model, in addition to the up-sampling and down-sampling of the synthetic minority over-sampling technique, often called SMOTE. The SMOTE is an oversampling where synthetic samples are produced for the minority class. The SMOTE, up-sampling and down-sampling process helps to overcome the overfitting problem by random generating new

instances into the NIDS dataset. Once the three sets of sub-sampled datasets are obtained, all three are consolidated to form a final input dataset for further feature selection and classification.

D. Feature Reduction (Feature Selection)

Feature selection is selecting a sub-sample of the most relevant features from the given NIDS datasets. Several methods are available to reduce the features. Chi-square test, principal component analysis, rank sum test and more are used on NIDS to reduce the dimensional of NIDS. Rank sum tests a non-parametric test with independent samples, assesses the correlation amongst the different variables and their distinct impact on classification accuracy. Thus, once selecting the set of significant features, it has been given as input to the proposed heterogeneous ensemble learning model for further training and classification.

E. Heterogeneous Ensemble Learning Model Design

Unlike classical standalone classifier-based anomalies detection or NIDS solutions, in this paper, we have designed a robust heterogeneous ensemble learning model with state-of-art high-performing machine learning algorithms such as Naïve Bayes, J48, k-NN, and RF. Noticeably, the above-stated machine learning algorithm's selection was based on their respective performances towards anomalies detection or allied IDS or NIDS solutions. Since the considered algorithms belong to the different paradigms such as J48 is an association mining model, while k-NN is a clustering concept, while RF is an ensemble model, the consideration of these all algorithms altogether constitute a heterogeneous ensemble learning environment, where each classifier acts as a base classifier to achieve a common consensus towards anomalies detection or NIDS solution.

F. Maximum Voting Ensemble (MVE) Model

To form an ensemble structure, all classifiers are executed over the same dataset and predict each connection traffic as an anomaly (label as "1") or normal traffic (label as "0"). Thus, obtaining each-connection outputs (i.e., label) of each node-edge values and corresponding label "Maximum Voting Ensemble (MVE) was executed that obtained the consensus for each connection traffic. Traffic or connection with higher 1's was identified as anomalies, while the traffic connection with higher 0s were classified as normal traffic. In MVE, each base machine learning classifier makes a prediction and votes for each sample. The sample class which has the highest votes is the final prediction for the output class. Thus, the proposed consensus-based model performed the identification of each connection as anomaly or normal traffic. Being a consensus-based approach, the reliability of the proposed heterogeneous ensemble model can be superior to the classical standalone classifier-based results.

IV. RESULTS AND DISCUSSIONS

To assess the performance of the proposed NIDS model (Fig.1-1.), the overall algorithms were developed using the WEKA tool, where it was simulated on the central processing unit (CPU) encompassing 64-bit Windows 10 machine with 16G RAM and i7-1.99GHz processor. To cross-examine the performance of both base-classifiers as well as the proposed heterogeneous ensemble model, we obtained aforesaid performance parameters over the different datasets. The performance assessment involves assessing true positive rate, false-positive rate, area under receiver operating characteristics (AUC).

A. Experiment-1 KDD'99 Datasets

In this section, the proposed NIDS model was simulated with the KDD'99 dataset, whose specifications are given in the previous sections. To assess whether the proposed heterogeneous ensemble model exhibits superior over the existing base-classifiers (standalone classifier), we obtained performance outputs by each base classifier as well the proposed Heterogeneous ensemble-based NIDS model. Moreover, realizing the fact that the majority of existing NIDS models merely apply binary classification, though the data used encompassed multiple attack types, signifying the possibility towards multi-class classification. Considering this fact, we performed both binary classification and multi-class classification, whose simulated results are given in Table 1.1 and Table 1.2, respectively. Trained and tested using varied machine learning models as base classifiers as well as proposed heterogeneous ensemble learning-based anomaly classifier.

TABLE 1. 1
PERFORMANCE ON KDD'99 FOR BINARY CLASS CLASSIFICATION

Classifiers	TPR	FPR	AUC
J48	0.961	0.011	0.999
k-NN	0.938	0.011	0.947
Naïve Bayes	0.917	0.028	0.981
SVM	0.923	0.019	0.992
Random Forest	0.936	0.016	0.999
Bagging	1	0	1
AdaBoost	1	0	1
*Proposed Heterogeneous Ensemble	**1**	**0**	**1**

TABLE 1. 2
PERFORMANCE ON KDD'99 FOR MULTI-CLASS CLASSIFICATION

Classifiers	TPR	FPR	AUC
J48	0.941	0.015	0.998
k-NN	0.925	0.140	0.910
Naïve Bayes	0.915	0.035	0.98
SVM	0.916	0.015	0.99
Random Forest	0.931	0.02	0.996
Bagging	0.961	0.111	0.998

AdaBoost	0.965	0.11	0.999
*Proposed Heterogeneous Ensemble	**0.988**	**0.007**	**0.997**

Observing results in Table 1.1, it can easily be found that unlike state-of-art standalone machine learning algorithms, our proposed Heterogeneous ensemble model exhibits near 100 % TPR , which is more than any other base classifier. Though, AdaBoost ensemble model too has performed relatively satisfactorily in terms of precision; however, being consensus-based approach, the proposed model seems more justifiable and convincing. Among all base classifiers, Naïve Bayes algorithm was found low-performing, in terms of TPR (91.5%) performance. Observing overall performance towards Binary decision in NIDS, the performance by our proposed Heterogeneous ensemble-based model seems more convincing and generalizable.

Considering multi-class classification ability by proposed model, the simulated results (see Table 1.2) reveals that similar to the binary class classification performance, the proposed Heterogeneous ensemble-based NIDS model achieves the highest TPR of 98.8%,. Moreover, false positive performance by the proposed model too was found significantly small, making the proposed Heterogeneous ensemble-based NIDS solution more reliable towards real-world intrusion detection purposes.

B. Experiment-2 NSL-KDD Datasets

The NSL-KDD is an improvised dataset with more non-linear connection patterns and sophisticated attacks conditions, making NIDS training more suitable for realistic anomaly identification. In this experiment, the machine learning algorithms were executing over the NSL-KDD dataset containing 42 features. Similar to the previous experiment

(Experiment-1), with same initial processing setups, different machine learning models were executed as standalone classifier, while the proposed Heterogeneous ensemble model embodies each connection-pattern classification and allied voting information to estimate consensus for final classification. The confusion metrics outcomes for both binary classification and multi-class attack classification are given in Table 1.3 and 1.4, respectively.

TABLE 1. 3
PERFORMANCE ON NSL-KDD FOR BINARY CLASSIFICATION

Classifiers	TPR	FPR	AUC
J48	0.815	0.146	0.841
k-NN	0.809	0.140	0.926
Naïve Bayes	0.761	0.198	0.908
SVM	0.851	0.098	0.969
Random Forest	0.805	0.155	0.959
Bagging	0.826	0.153	0.928
AdaBoost	0.784	0.174	0.903
*Proposed Heterogeneous Ensemble	**0.945**	**0.012**	**0.963**

Considering binary class classification that is, intrusion or malicious traffic detection, the proposed Heterogeneous ensemble-based NIDS model achieves the highest TPR of 94.5% and lower FPR of 0.012. Multi-class classification efficacy by the proposed Heterogeneous ensemble model, one can easily observe that unlike other state-of-art methods, our proposed Heterogeneous ensemble model exhibits superior in terms of highest TPR (92.5%), lowest FPR (0.016) (see Table 1.4).The overall result confirms the robustness of the proposed Heterogeneous ensemble-based NIDS system for real-world anomalies detection.

TABLE 1. 4
PERFORMANCE ON NSL-KDD FOR MULTI-CLASS CLASSIFICATION

Classifiers	TPR	FPR	AUC
J48	0.781	0.154	0.840
k-NN	0.799	0.143	0.912
Naïve Bayes	0.755	0.199	0.905
SVM	0.823	0.112	0.950
Random Forest	0.795	0.165	0.949
Bagging	0.806	0.173	0.918
AdaBoost	0.754	0.184	0.901
*Proposed Heterogeneous Ensemble	**0.925**	**0.016**	**0.943**

C. Experiment-3 UNSW-NB15 Dataset

In this experiment, the UNSW-NB15 dataset was taken into consideration for the NIDS process. The results obtained towards the binary classification, that is, Normal traffic or anomalies connection, the proposed ensemble learning-based concept was found superior over other state-of-art methods. The proposed Heterogeneous ensemble-based NIDS model exhibited the highest TPR of 100%, which is more than all base classifiers individually. Additionally, the FPR (0%) by the proposed model was lower than the standalone classifier. In the same way, the false-positive rate was found to be lower than the other methods for both binary classification (see Table 1.5) as well as multi-class classification (see Table 1.6). For multi-class classification as well, the proposed NIDS model with heterogeneous ensemble learning ability exhibited the highest TPR of 0.931, with 0.006 FPR signifying reliability towards the NIDS solution.

TABLE 1. 5
PERFORMANCE ON UNSW-NB15 FOR BINARY CLASSIFICATION

Classifiers	TPR	FPR	AUC
J48	1	0	1
k-NN	1	0	1
Naïve Bayes	0.911	0.021	0.968
SVM	0.882	0.073	0.905
Random Forest	0.996	0.003	1
Bagging	1	0	1

AdaBoost	1	0	1
*Proposed Heterogeneous Ensemble	**1**	**0**	**1**

TABLE 1. 6
PERFORMANCE ON UNSW-NB15 FOR MULTI-CLASS CLASSIFICATION

Classifiers	TPR	FPR	AUC
J48	0.864	0.017	0.947
k-NN	0.744	0.018	0.893
Naïve Bayes	0.623	0.035	0.956
SVM	0.810	0.233	0.788
Random Forest	0.849	0.031	0.966
Bagging	0.874	0.019	0.961
AdaBoost	0.85	0.02	0.979
*Proposed Heterogeneous Ensemble	**0.931**	**0.006**	**0.945**

Observing overall results, it can easily be found that the proposed NIDS model using ensemble learning ability exhibits superior, which could be contributed due to high-accurate base classifiers consensus or voting. Undeniably, being an MVE consensus-based approach, the proposed NIDS model's reliability or accuracy is higher and more generalizable as a real-world solution. Though it requires exploiting the consensus by each base-classifier it takes relatively higher training time; however, parallel execution of the different base-classifiers makes the proposed system more time-efficient. Therefore, it provides optimally reliable and accurate performance, even at the cost of computational time addition. It makes the proposed system suitable for real-world NIDS solution.

V. CONCLUSION

Though a number of existing literatures apply machine learning method(s) for anomalies detection or classification; however, their respective generalization over different datasets remains questionable. Different classifiers show different performances or results over the same dataset. It raises the question of their acceptability as a universal NIDS solution. Unlike classical methods, the use of multiple network traffic data such as NSL-

KDD, KDD99 and UNSW-NB15 strengthened the proposed data-driven NIDS model to detect both known as well as unknown traffic patterns. Subsequently, the proposed model applied rank-based features and thus achieves higher computational efficacy with better training and knowledge generation. Thus, employing a maximum voting ensemble or consensus-based approach, the proposed Heterogeneous ensemble model predicts each pattern or node as anomalies or normal, hence increasing the accuracy of the proposed model. Moreover, it performed multi-class classification as well, with higher true-positive rate, higher AUC and high-accuracy. Noticeably, the proposed ensemble learning model achieved better performance than the comprising base-classifiers (as standalone classifier). The higher detection rate, with the higher true positive rate , low false positive rate and AUC confirms suitability of the proposed model towards real-world NIDS applications.

CHAPTER 2. EFFICACY OF MACHINE LEARNING-BASED CLASSIFIERS FOR BINARY AND MULTI-CLASS NETWORK INTRUSION DETECTION

Abstract—The internet-based services undoubtedly led the worldwide revolution with exponential growth, but security breaches resulted in personal digital asset losses, which need a comprehensive cybersecurity solution. Traditionally, signature-based network intrusion detection is employed to capture attributes of normal and abnormal traffic in a network, but it fails to detect the zero-day attack. The machine learning-based approach is attractive among various known NIDS methods to circumvent the shortcoming because the machine learning-based approach can efficiently analyze the big network traffic data and efficiently detect the zero-day attack. The imbalanced NIDS dataset does not provide better performance on practical implementation scenarios. Reducing the number of target classes into a new target class creates a balanced NIDS and improves classifier performance. In this paper, we present the efficacy of several machine learning algorithms, including Random Forest (RF), J48, Naïve Bayes, Bayesian Network, Bagging, AdaBoost, and Support Vector Machine (SVM) using network logs traffic (KDD99, UNSW-NB15, and CIC-IDS2017) using WEKA. This paper examined the impact of changing the number of output classes of the publicly available network intrusion datasets on sensitivity (True Positive Rate), False Positive Rate (FPR), Area under the ROC curve (AUC), and incorrectly identified percentage. Interestingly, the efficiency of these classifiers has increased, adding strongly correlated features to the target classes. The experiment results reveal that the machine learning classifiers' performance improved

when the number of target classes decreased. The addition of a highly correlated feature to the output class increases the performance of the classifiers.

Keywords—CIC-IDS2017, imbalanced dataset, KDD99, machine learning, network intrusion detection system, UNSW-NB15.

I. INTRODUCTION

An increase on the Internet and technology usage has increased cyberattack risk: losing data, personal information, and more. Cyber attackers can use different attacks to compromise the CIA (Confidentiality, Integrity, and Availability) triad. An Intrusion detection system (IDS) is used to inspect networks and host activity to identify suspicious traffic. A system to detect attacks related to the host is called host IDS; likewise, the network intrusion detection system (NIDS) detects the network devices' attacks. The host devices are the data sources for the host-based IDS. Security information and event management (SIEM) is used to analyze the traffic for the host-based intrusion. Network-based IDS, network devices such as routers, switches, and host terminals switch are used to collect the traffic data to analyze the networks' malicious activities. Most intrusion detection systems consist of network sensors, central monitoring systems, report analysis, database and storage components, and response boxes. The IDS can be a hardware or software device used to detect malicious activities inside the host and computer networks. There are two categories of intrusion detection methods: signature-based intrusion detection systems (SIDS) and anomaly-based intrusion detection systems (AIDS) [1].

SIDS is a traditional method where malicious activities are detected based on pattern matching with external attacks, including knowledge-based detection and misuse detection. The SIDS matches the current signature with the signature database's previous signature and

checks for suspicious activities. The main drawback of the SIDS is that it fails to detect zero-day attacks as those new types of attack signatures are not included in the SIDS signature database. The commonly used IDS are Snort and NetSTAT. AIDS overcomes the SIDS's drawbacks by modeling normal behaviors using machine learning, statistical-based, or knowledge-based methods [1]. Anomaly-based detection can also produce false results caused by changes in user habits.

The fuzzy C-means clustering techniques combined with principal component analysis and the genetic algorithm used to optimize the provided network-based IDS produced poor accuracy on R2L (90.2%) and U2L (23.8%) for KDD99 [33]. Simultaneously, the performance using genetic search dimensionality reduction on KDD99 showed that probing, U2R, and R2L yield a true positive rate of 72.4%, 9.3%, and 2.6%, respectively [34]. While [35] used a fuzzy genetic search algorithm and received 97.97% maximum performance, even they used feature reduction on the KDD99 dataset. For the decision tree, U2R showed a poor performance of 88.33% on KDD99 using a flexible neural tree and particle swarm optimization method with different feature subsets [36]. The number of features and feature subsets obtained by dimensionality reduction and feature reduction does not improve classification accuracy.

The detection rate using a genetic search algorithm to select the features subset on KDD99 with Synthetic Minority Oversampling Technique (SMOTE) using random forest classifier, the U2R was high of 82.7%. The SMOTE method was used on NSL-KDD and UNSW-NB15 datasets to remove the skewness [37]. Other authors also tried to improve the classification accuracy for the KDD99 dataset using double-layer detection and stacking ensemble classifiers [38] and the cluster-based under-sampling method [39]. The ensemble

method improves normal traffic performance while it shows poor performance on rare attacks. Similarly, the SMOTE techniques generate redundant instances for minor classes. They are under-sampled for the major class traffic, which is not perfect for the NIDS to produce an accurate result on the real network traffic and attacks.

Deep learning methods for NSL-KDD and UNSW-NB15 datasets found the detection rate on U2R and R2L with the precision of 85.58% probe and 59.52% U2R attacks on NSL-KDD with overall accuracy on UNSW-NB15 is 86.5% [40]. An in-depth learning approach was combined with a sparse autoencoder and SVM for NSL-KDD and CICIDS 2017 to improve the performance [18]. The ANN used on the UNSW-NB15 on the NIDS dataset showed the highest precision recorded, with 88% for binary classification [41]. The autoencoder and deep learning on UNSW-NB15 provided the highest accuracy of 98% [42], and CICIDS2017 [43] TPR of 99.8%. Unfortunately, the deep learning system requires a large dataset and demands higher computing power and expensive GPUs, which increases the end-user cost.

Hence, we proposed a simple, efficient method where we regroup the rare attacks into new classes to reduce the number of target classes and increase the number of instances in the target class without interpolating the traffic of the imbalance NIDS datasets (KDD99, NSL-KDD, UNSW-NB15, CICIDS2017) to provide a better intrusion detection rate in a real scenario NIDS. The different classifiers such as Naïve Bayes, Bayesian Network, Random Forest, J48, AdaBoost, and Bagging are tested to perform binary and multiclass classification.

The main contributions of this research work are:

1) Reducing the number of classes increases the performance of machine learning classifiers.
2) Deriving highly correlated features with the output class improves the machine learning model's performance on intrusion datasets.
3) Benchmarking different machine learning algorithms on different intrusion datasets also provides a similar insight into our approach's efficacy.

The remainder of the paper is as follows. Section II describes the methodology of our proposed NIDS approach. Section III illustrates the results and discussion, while Section IV concludes this research work.

II. METHODOLOGY

Machine learning-based NIDS performance is affected by the size of the training and testing dataset, number of target classes, nature of the dataset, number of features, and machine learning algorithms used. The network traffic is collected, examined, and processed using a network analyzer inside the network to create the NIDS dataset. The preprocessed data is used to train and test the selected algorithms. Performance metrics are recorded to compare and analysis the model.

A. Data Description

This research uses publicly available NIDS datasets (*.CSV), the KDD99, UNSW-NB15, and CICIDS2017. A brief description of these datasets is as follows.

1) ***KDDCup99:*** The KDDCup99 dataset, also referred to as KDD99 data [30], was developed by executing the 1998 DARPA intrusion detection challenge dataset by processing TCP dump data. This data is available in full dataset volume and 10% data volume. It has 41 features with five distinct attack classes: Normal, DoS, Probe, R2L, and U2R (see Table 2.1).

TABLE 2. 1
TRAINING AND TESTING DATA ON KDD [6]

Attack Categories	**Train**	**Test**
Normal	812,814	60,593
DoS	247,267	229,853
Probe	13,860	4,166
R2L	999	16,189
U2R	52	228
Total	1,074,992	311,029

2) ***UNSW-NB15:*** The Australian Centre for Cyber Security (ACCS) cybersecurity research team created the UNSW-NB15 dataset [32] to solve issues with the KDD99 dataset. The data used in this research comprises 42 features. The distribution of the data used in this research work is as per Table 2.2

TABLE 2. 2
UNSW-NB15 DATASET [6]

Class	**Training**	**Testing**
Normal	56,000	37,000
Fuzzers	18,184	6,062
Analysis	2,000	677
Backdoors	1,746	583
DoS	12,264	4,089
Exploits	33,393	11,132
Generic	40,000	18,871
Recon	10,491	3,496
Shellcode	1,133	378
Worms	130	44
Total	175,341	82,332

3) ***CICIDS2017*:** The University of New Brunswick's Canadian Institute for Cybersecurity released the CICIDS2017 [44] dataset according to the framework defined in [45] to solve previous dataset's shortcomings. Raw data in the form of PCAP files are provided together with 79 features in CSV files. Network traffic was recorded over 5 days from 9 a.m. Monday, July 2017, to 5 p.m. on Friday, July 2017, resulting in a total of 2,830,743 instances. The Wednesday working hours and Thursday morning web attacks network traffic used during this research work are as seen in Table 2.3.

TABLE 2. 3
CICIDS2017 WEDNESDAY AND THURSDAY M. TRAFFIC [16]

Wednesday Traffic		Thursday Morning Traffic	
DoS GoldenEye	10,293	WA_brute_force	1,507
DoS Hulk	231,073	WA_Sql_injection	21
DoS Slowhttptest	5,499	WA_XSS	652
DoS Slowloris	5,796		
Heartbleed	11		
Benign	44,0031	Benign	168,186

B. Data preprocessing

We converted the dataset into Attribute-Relation File Format (*. arff) from *.CSV file format for WEKA compatibility. The data preprocessing includes feature selection, normalization, handling missing values, null values, and redundant instances in the NIDS datasets. The proposed method reduces the number of classes by regrouping the additional intrusion traffic into different classes. The grouping of different target classes makes the NIDS dataset more balanced. The detailed feature processing for each dataset is as follows:

1) ***KDDCup99 preprocessing:*** Here, we created a binary class dataset from a multiclass by grouping all attacks into a single class and remaining grouped into a normal class. The resultant dataset becomes a binary class with a 'normal' and 'attack' class.

2) ***UNSW-NB15 Preprocessing for Multiclass Experiment:*** The multiclass (10) UNSW-NB15 dataset is converted into binary class NIDS by grouping all the attack classes (see Table 2.2 for attack list) into a single class and the rest are in normal classes for the binary data set experiment. Again, we converted the UNSW-NB15 multiclass with 10 output classes dataset into UNSW-15 multiclass with 6 output classes by combining attack types of worms, shellcode, backdoor, analysis, and recon attacks grouped into a new attack class, keeping other classes unchanged. The grouping of the attack classes with a low number of instances into new single classes reduces NIDS skewness. It improves the detection rate because this contributes to the dataset becoming a more balanced form.

3) ***CIC-IDS2017 Preprocessing for Multiclass experiment:*** Conversion of multiclass (6) to binary class is done by grouping all attack types into a single target class. We create the multiclass (4) NIDS dataset for Wednesday traffic (see Table 2.3 for attack list) by grouping Heartbleed, DoS Slowloris, and Dos Slowhttptest into a single attack class. Since the given dataset is an imbalanced one, regrouping the different attacks into a new class improves imbalance NIDS performance. Also, we regroup WA_sql_injection and WA_XSS into a single class to form the multiclass (3) CICIDS2017 NIDS to perform the experiment and compare the result between different multiclass experiments. The regrouping is based on the lower number of attacks into a new attack class.

4) ***UNSW-NB15 Preprocessing for Feature Engineering:*** The UNSW-NB15 dataset contains 45 features. Among them, #44 (attack_cat) and #45 (label) are important features to compare classifiers' performance. The "attack_cat" features represent the nine types of attacks included in the UNSW-NB15 dataset. These nine attacks are labeled as "abnormal" and good traffic as "normal" using the UNSW_NB15 dataset attacks table. The label (feature #45) contains two values, "abnormal" and "normal," created based on feature #44 (attack_cat). For binary class, feature number #45 (label) was assigned as the output class, providing the "normal" and "abnormal" as the output class. Similarly, for multiclass UNSW-NB15, feature number #44 (attack_cat) was assigned as the output class, providing 10 different classes: Normal, Reconnaissance, Shellcode, Exploit, Fuzzers, Worm, DoS, Backdoor, Analysis, and Generic. Similarly, for the binary class UNSW-NB15 dataset, feature number #45 (label) was assigned as an output class, and features number #1 (id) and feature number #44 (attack_cat) were from the dataset. As a result, the training and testing dataset became the binary class with 43 features. For multiclass, feature number #44 (attack_cat) was assigned as an output class, and we removed feature number #1 (id) and feature number #44 (label) from the UNSW-NB15 dataset. Then, the resulting dataset became a multiclass dataset with 43 features.

C. Data splitting

The train test split ratio is not defined in the literature. The researchers [46] adopt 60% of training and 40% of test data for building the machine learning model. We used the same ratio for this research by creating separate files for training and test data.

D. Machine Learning Algorithms

The selection of a machine learning model is performed from different categories. The Naïve Bayes (NB) and Bayesian Networks are chosen as the probabilistic model and are dominantly used in the previous NIDS research. The J48 and RF are decision tree-based classifiers. The decision tree-based machine learning classifiers used a large number of trees to make the decision and produce the final result. Hence, the tree-based machine model was also used to perform the experiments. Along with this, we used ensemble classifiers to measure the NIDS classification performance using Bagging and AdaBoost algorithm. We used the ensemble classifiers because these classifiers are robust for the imbalance NIDS dataset. The selection of all different machine learning classifiers itself provides a detailed study of which categories classifiers provide high classification performance.

The hyperparameters and other experimental configurations on the WEKA are the batch size used for all the machine algorithms is 100. For the J48 classifier, we choose the confidential factor 0.25 for subtree raising pruning and choose onefold of the dataset for pruning and 2-fold of the dataset for growing the tree. There is the minimum number of instances per leaf, i.e., 2. The algorithm used minimum description length (MDL) correction to split numeric attributes. The RF classifier uses single execution slots (threads) to construct the ensemble, and the single bag consists of all the training data. The maximum depth of the tree used is 0, which refers to the unlimited. For BaysianNet, for finding the conditional probability tables, the simple estimator α equal to 0.5. It uses search algorithms where the order of the variables restricts hill-climbing algorithms. The Bayes score type is used to judge the quality of the network. The initial network used for structure learning is the Naïve Bayes network. For Adaboost, we randomly select the base classifiers, which are

decision stump, decision tree, and random forest, for 10 iterations. The weight threshold for weight pruning is 100. For bagging, Single slots (thread) are used for constructing the ensemble. All the training dataset is performed as a single bag. For the fast decision tree learner (as a base classifier), we use the minimum total weight of the instance in a leaf as 2, where pruning does not apply and uses any maximum tree depth. The minimum proportion of the variance on all the data present at a node for splitting in a regression tree is 0.001, and the initial class counts 0.0 for a 3-fold.

E. Model Evaluation

The performance metrics are recorded and compared between binary and multiclass NIDS datasets in terms of true positive rate (TPR), false-positive rate (FPR), incorrectly classified percentage, and area under the ROC (AUC). TPR is also called detection rate or sensitivity which is the ratio of correctly predicted instances and total number of instances. The FPR is the ratio between the number of normal instances that are classified incorrectly as attacks and the total number of normal traffic instances. The incorrectly classified percentage is obvious to define, which provides the how much percentage of total data that is incorrectly classified. The AUC is the Area under the ROC, which provides the relation between TPR versus FPR for the given classifiers. The highest value of TPR, lowest value of FPR, and highest values of the AUC rated the machine learning models during the performance evaluation.

The proposed NIDS model's overall implementation is as per the following pseudocode.

Pseudocode for Proposed NIDS for WEKA	
1	FOR $\forall u;$ $u = [KDD99, UNSW-NB15, CICIDS2017]$
2	Regroup target classes (see Data Preprocessing Section B)
3	Build a sampling dataset of features.
4	Split training and testing dataset a 60:40 ratio
5	FOR $\forall v;$ $v = [NB, J48, RF, BayesinNet, Bagging, Adaboost]$
6	Build model m_{uv} using WEKA; m= model id.
7	Calculate performance metrics.
8	END
9	Compare the performance metrics. $\forall m$
10	END
11	Terminate

III. RESULT AND DISCUSSION

The experiments were simulated using WEKA installed on the central processing unit encompassing a 64-bit Windows 10 machine with 16G RAM and an i7-1.99GHz processor. The performance of the models was compared using various performance metrics including, total program run time, detection ratio, and precision. The higher TPR represents the model is performing well.

A. Experiment-1 KDD'99 Datasets

The binary and multiclass performance results are shown in Table 2.4 and Table 2.5, respectively. The high value of the binary class NIDS shows that the given machine learning models show higher classification performance.

TABLE 2. 4
PERFORMANCE ON KDD'99 FOR BINARY CLASSIFICATION

Classifiers	TPR	FPR	AUC	Incorrectly Classified (%)
NB	0.917	0.028	0.981	8.24
J48	0.961	0.011	0.999	3.914
RF	0.936	0.016	0.999	6.444
BayesinNet	0.923	0.015	0.995	7.74
Bagging	1	0	1	0
Adaboost	1	0	1	0

TABLE 2. 5
PERFORMANCE ON KDD'99 FOR MULTICLASS CLASSIFICATION

Classifiers	TPR	FPR	AUC	Incorrectly Classified (%)
NB	0.915	0.035	0.98	8.347
J48	0.941	0.015	0.998	5.35
RF	0.931	0.02	0.996	6.871
BayesinNet	0.916	0.019	0.99	8.436
Bagging	0.961	0.111	0.998	3.914
Adaboost	0.965	0.11	0.999	3.945

By observing the results in Table 2.4 for the KDD99 dataset, it is found that the TPR for binary class NIDS is higher than the multiclass NIDS. Comparing those results in Table 2.4 and 2.5, the binary class performance is higher than the multiclass performance, which is due to the regrouping of the classes and removing the skewness in the dataset. The performance result shows that Bagging and Adaboost algorithms have better performance than the other machine learning models because ensemble models are combined with multiple models, which are robust.

B. Experiment-2 UNSW-NB15 Dataset

The performance metrics for both binary classifications and multiclass attack classifications are given in Table 2.6. The Bagging and AdaBoost ensemble classifiers shown in Table 2.6 have 100% TPR for the binary class NIDS. The TPR for all classifiers is low for multiclass (10) as compared to multiclass (6). The ensemble learning-based concept was found superior over other state-of-the-art methods for binary classification based on higher TPR. This clearly shows that the classifier's TPR values are improved when the number of classes is reduced. The overall comparison in Table 2.6 shows that the performance for every machine learning algorithm increases with a decrease in the number of class sizes.

C. Experiment-3 CICIDS2017 Dataset

Moreover, the bagging and AdaBoost ensemble model exhibit 100% of TPR on binary KDD'99 NIDS. The experimental results reveal that the binary class classification performances with the highest accuracy of 100%. Table 2.7 and Table 2.8 experimental result shows that the performance of binary class CIC-IDS2017 Wednesday and Thursday NIDS obtained 100% TPR for decision tree-based classifiers and ensemble classifiers. As in KDD'99 and UNSW-NB15 NIDS, classifiers' performance increased when the number of target classes decreased on CIC-IDS2017 NIDS.

D. Experiment-4 UNSW-NB Dataset (feature engineering)

This experiment-4 is implemented on UNSW-NB15 multiclass and binary class NIDS. Interestingly, It is found that the attributes play an important role in improving the performance of UNSW-NB15 NIDS. The last two attributes are used to determine whether the NIDS is binary or multiclass: one contains the multiclass target and the other binary class target value. Preserving the features that are used for NIDS to create the target binary or multiclass increased the performance of the machine learning models. The details about the name of the features are explained in the dataset information and data preprocessing section for UNSW-NB15 dataset. The literature reflected that the researchers used either one feature during the classification of UNSW-NB15 network traffic.

Those features are highly correlated with each other because the binary target classes are derived directly based on the number of attacks grouped into single classes and keeping the good traffic in the normal class. Similarly, the multiclass target classes are directly related to the type of attack classes that reside in the NIDS dataset. The conversion from

multiclass to binary made the simple, but preserving those features in the NIDS dataset increased the performance of the machine learning classifiers.

TABLE 2. 6.
PERFORMANCE OUTPUTS WITH UNSW-NB15 DATASET BINARY AND MULTICLASS CLASSIFICATION

Performance of Binary Class UNSW-NB15					Multiclass Class-6 UNSW-NB15				Multi Class-10 UNSW-NB15			
Classifiers	TPR	FPR	AUC	Incorrectly Classified (%)	TPR	FPR	AUC	Incorrectly Classified (%)	TPR	FPR	AUC	Incorrectly Classified (%)
NB	0.911	0.021	0.968	8.8593	0.684	0.029	0.96	31.6	0.623	0.035	0.956	37.7119
J48	1	0	1	0	0.903	0.014	0.976	9.726	0.864	0.017	0.947	13.578
RF	0.996	0.003	1	0.3887	0.9	0.016	0.985	9.9826	0.849	0.031	0.966	15.0743
Bayesin Net	0.913	0.098	0.969	8.7014	0.688	0.128	0.961	31.16	0.624	0.25	0.957	37.6399
Bagging	1	0	1	0	0.903	0.013	0.987	9.6565	0.874	0.019	0.961	12.5808
Adaboost	1	0	1	0	0.896	0.011	0.981	10.4017	0.85	0.02	0.979	15.0221

TABLE 2. 7
PERFORMANCE ON CICIDS2017 BINARY AND MULTICLASS CLASSIFICATION FOR THURSDAY DATA

Multiclass-4 CIC-IDS2017_Thursday					Multiclass-3 IC-IDS2017_Thursday				Binary classCIC-IDS2017Thursday			
Classifiers	TPR	FPR	AUC	Incorrectly Classified (%)	TPR	FPR	AUC	Incorrectly Classified (%)	TPR	FPR	AUC	Incorrectly Classified (%)
NB	0.972	0.025	0.993	2.778	0.975	0.018	0.995	2.516	0.982	0.004	0.999	1.755
J48	0.99	0.011	0.995	0.394	0.996	0.01	0.997	0.397	1	0.005	0.999	0.028
RF	0.994	0.033	1	0.467	0.995	0.031	1	0.4648	1	0.03	1	0.042
Bayesin Net	0.953	0.03	0.994	4.658	0.965	0.025	0.999	3.485	0.983	0.015	0.999	1.7
Bagging	0.995	0.029	1	0.4181	0.996	0.019	1	0.4268	1	0.017	1	0.0276
Adaboost	0.994	0.012	1	0.4285	0.996	0.01	1	0.4095	1	0.005	1	0.0104

TABLE 2. 8
PERFORMANCE ON CICIDS2017 BINARY AND MULTICLASS CLASSIFICATION FOR WEDNESDAY DATA

Multiclass-6 CIC-IDS2017_Wednesday					Multiclass-4 CIC-IDS2017_Wednesday				Binary class CIC-IDS2017_Wednesday			
Classifiers	TPR	FPR	AUC	Incorrectly Classified (%)	TPR	FPR	AUC	Incorrectly Classified (%)	TPR	FPR	AUC	Incorrectly Classified (%)
NB	0.918	0.019	0.987	8.171	0.926	0.019	0.998	7.372	0.94	0.015	0.999	5.959
J48	0.998	0.001	1	0.052	1	0	1	0.045	1	0	1	0.036
RF	0.997	0.001	1	0.067	0.998	0.001	1	0.0622	0.999	0.001	1	0.054
Bayesin Net	0.991	0.009	0.997	0.982	0.995	0.009	0.999	0.612	0.997	0.006	0.999	0.524
Bagging	1	0	1	0.0481	1	0	1	0.0445	1	0	1	0.0347
Adaboost	1	0	1	0.0441	1	0	1	0.0325	1	0	1	0.0231

The performance Table 2.9-2.12 shows that the TPR is higher when both attributes (target class with binary classes and target class with multiclass) are present on the UNSW-NB15 datasets. The binary class with 45 features UNSW-NB15 NIDS on Table 2.9 has 100% TPR for ensemble classifiers. In contrast, the binary class with 43 features UNSW-NB15 NIDS on Table XI shows a lower detection rate for ensemble classifiers (86 % TPR). Similarly, in multiclass 10 UNSW-NB15 NIDS, the TPR is lower in Table 2.12 for 43 features NIDS than for 45 features NIDS in Table 2.11 because those two attributes are highly correlated.

TABLE 2. 9
PERFORMANCE ON BINARY UNSW-NB15 WITH 45 FEATURE

Classifiers	TPR	FPR	AUC	Incorrectly Classified (%)
NB	**0.911**	**0.021**	**0.968**	**8.8593**
J48	**1**	**0**	**1**	**0**
RF	**0.996**	**0.003**	**1**	**0.3887**
BayesinNet	**0.913**	**0.098**	**0.969**	**8.7014**
Bagging	**1**	**0**	**1**	**0**
Adaboost	**1**	**0**	**1**	**0**

TABLE 2. 10
PERFORMANCE ON BINARY UNSW-NB15 WITH 43 FEATURE

Classifiers	TPR	FPR	AUC	Incorrectly Classified (%)
NB	0.832	0.195	0.929	16.7893
J48	0.87	0.154	0.915	12.9707
RF	0.874	0.15	0.98	12.5528
BayesinNet	0.833	0.195	0.93	16.7456
Bagging	0.868	0.157	0.982	13.184
Adaboost	0.868	0.156	0.972	13.1759

TABLE 2. 11
PERFORMANCE ON MULTICLASS UNSW-NB15 WITH 45 FEATURE

Classifiers	TPR	FPR	AUC	Incorrectly Classified (%)
NB	**0.623**	**0.035**	**0.956**	**37.7119**
J48	**0.864**	**0.017**	**0.947**	**13.578**
RF	**0.849**	**0.031**	**0.966**	**15.0743**
BayesinNet	**0.624**	**0.25**	**0.957**	**37.6399**
Bagging	**0.874**	**0.019**	**0.961**	**12.5808**
Adaboost	**0.85**	**0.02**	**0.979**	**15.0221**

TABLE 2. 12
PERFORMANCE ON MULTICLASS UNSW-NB15 WITH 43 FEATURE

Classifiers	TPR	FPR	AUC	Incorrectly Classified (%)
NB	0.605	0.046	0.937	39.4718
J48	0.753	0.037	0.915	24.7376
RF	0.756	0.028	0.963	24.3793
BayesinNet	0.634	0.045	0.938	37.8542
Bagging	0.76	0.032	0.952	24.0453
Adaboost	0.753	0.037	0.928	24.7376

IV. CONCLUSION

The primary goal of IDS is to detect network attacks. We find that reducing the number of classes increases machine learning classifiers' performance, that is, converting the multiclass dataset to a binary dataset. Also, we observed that keeping the highly correlated attributes with the target class, attribute with binary class values and attribute with multiclass values, increased the machine learning model's performance. Reducing the target class numbers contributed to creating a balanced dataset and eliminating complexity. Additionally, we prove that our approach is reliable and accurate and uses very low computational resources as compared to the deep learning method, making it suitable for real-world NIDS applications. The deep learning method requires high computing power where high random-access memory (RAM) and graphical processing units (GPU) are required to process the big NIDS dataset. Furthermore, we get similar insights on all NIDS datasets while benchmarking on different machine learning algorithms, proving our approach's efficacy.

Reducing the number of target classes into fewer target classes by regrouping the smaller attack classes reduces the imbalances in the NIDS data sets because the number of instances of the simulated attacks in the NIDS is always lower as compared to the known

attack classes. Similarly, preserving the features related to the target class, the feature that is used to deduce the target classes, also increases the performance of the machine learning classifiers for the given NIDS because these features are used to deduce the final target classes directly.

The selection of the three different groups of machine learning classifiers was also based on the probabilistic classifiers because these types of classifiers were dominantly used in previous days in NIDS research. The decision tree is based on predictive modelling approach-based machine learning models. The last categories are the ensemble-based machine learning models, which work based on the classifier during classification. The selection of the different machine learning models during this NIDS experiment concludes that the ensemble methods improved the performance and, hence, intrusion detection accuracy.

CHAPTER 3. EFFICACY OF BIDIRECTIONAL LSTM MODEL FOR NETWORK-BASED ANOMALY DETECTION

Abstract- The Internet is vital in daily applications such as education, health, business, etc. Increasing the usage of the Internet and technology also increases the risk. Cyber attackers can use technology to compromise the triad of the CIA (Confidentiality, Integrity, and Confidentiality). Malicious activities occur in our surroundings without our knowing it. Cyberattacks cannot be seen physically, though occurring to the Internet of things (IoT) devices, personal computers, laptops, and even the networking devices. Network anomaly detection is an efficient way of detecting malicious activities. Network-based anomaly detection captures and analyzes attributes of abnormal behavior in a network. Machine learning and deep learning-based approaches are attractive among various known methods for network anomaly detection because they can efficiently analyze big network traffic data for malicious activities and detect zero-day attacks. A Recurrent Neural Network (RNN) model is designed to recognize the sequential characteristics of data and then use the patterns to predict the coming scenario. In this research work, seven different optimizers (Nadam, Adam, RMSprop, Adamax, SGD, Adagrad, and Ftrl), epochs, batch size, and the ratio of training testing data size are analyzed for the Bidirectional Long Short-Term Memory (Bi-LSTM) network anomaly detection which provides the highest anomaly detection accuracy of 98.52% on the NSL-KDD binary dataset. The performance is compared using accuracy and F1-score metrics. Performance assessment regarding the accuracy and F1-score revealed that the proposed Bi-LSTM anomaly detection model exhibited better performance than the other existing anomaly detection methods.

Keywords— Bi-LSTM, deep learning, LSTM, network intrusion detection system, NSL-KDD, machine learning

I. INTRODUCTION

With the invention of information and technology, the most crucial information is transmitted in the form of bits from source to destination. The transmitted information can be voice, image, or data, containing banking information, personal information, and network traffic. Various tools or methods are available to detect and prevent intruders. Anomaly is a pattern in the dataset that does not fit into the usual behavior of the data, and some detection techniques are required to detect it. Outliers and anomalies are sometimes used interchangeably in the field of anomaly detection. Anomaly detection has numerous applications, including business, network intrusion detection, health monitoring systems, credit card fraud detection, and fault detection in critical information systems. Anomaly detection is important in cyber security for achieving solid protection against cyber adversaries. A system is considered secure if the three computer security principles of Confidentiality, Integrity, and Availability (CIA) are properly met [47]. An intrusion detection system is a method for monitoring and examining what is happening in a computer or network system to detect potential risks by evaluating how often CIA computer security guidelines are broken.

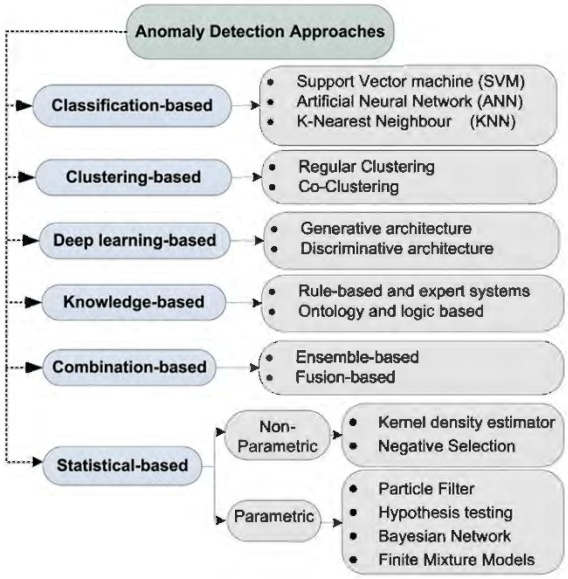

Fig. 3. 1. Taxonomy of anomaly detection [48]

There are two categories of intrusion detection methods: signature-based intrusion detection systems (SIDS) and anomaly-based intrusion detection systems (AIDS). Anomaly detection systems are classified into two categories based on the sources: network-based and host-based intrusion detection systems. Anomaly detection techniques utilize labels to identify whether the data is normal or anomalous. There are three different anomaly detection techniques such as supervised, unsupervised, and semi-supervised anomaly detection methods. AIDS overcomes the SIDS's drawbacks by modeling normal behaviors using machine learning, statistical-based, or knowledge-based methods. The different anomaly detection approaches are listed below in Fig. 3.1 [48].

Deep learning can extract better representations for creating efficient anomaly detection models. The traditional machine learning-based network anomaly detection algorithms are more suited for small datasets and are mostly performance dependent on how the feature engineering is implemented. The split ratio is one of the dominant elements influencing the performance of traditional machine learning-based anomaly detection methods. The traditional ML methods are simple and have low resource consumption. Still, for huge datasets and large features, poorly performed and traditional ML cannot be worked on computer vision, natural language processing, image translations, etc. The Convolutional Neural Network (CNN) is mostly used in image datasets where the lower layer’s neurons do the feature reduction in the network, usually identifying important small-scale features, such as boundaries, corners, and intensity differences. Then in higher layers, the network combines the lower-level features to form more complex features such as simple shapes, forms, and partial objects. And on the final layer, the network combines the lower features to produce the output or classification results. LSTM works differently than a CNN because

an LSTM is usually used to process and make predictions given data sequences. RNNs were designed to retain long-range information so that the information is remembered and not lost in a long sequence. BiLSTM adds one more LSTM layer, reversing the information flow direction and overcoming the vanishing gradient problems.

The deep learning method overcomes the problems in traditional ML, such as being suited for huge datasets and large numbers of features. The performance of the deep learning-based anomaly detection algorithm depends on the number of neurons, number of hidden layers, types of activation function, number of samples (batch size), and epochs (iterations) during DL model training and testing. Selecting those hyperparameters, training testing data ratio, and architecture of neural network in deep neural networks is vital in increasing the detection accuracy of network anomaly detection systems.

II. RELATED WORK

The volume of big data is growing daily, so the traditional machine learning algorithm cannot be performed well and needs intensive feature engineering tasks. Deep learning greatly improves detection performance. Still, the nature of the dataset used in network anomaly detection (balanced and unbalanced), the hyperparameters on deep neural networks, training, testing data size, and neural network architecture play a vital role in detecting the anomaly.

Authors [49] implemented the Bi-LSTM model to overcome the extensive feature engineering task required for traditional machine learning-based anomaly detection. Also, data augmentation used during data preprocessing on rare attacks (U2R, R2L) was applied to create the balanced NSL-KDD dataset resulting in the accuracy and F1 scores better than other comparison methods, reaching 90.73% and 89.65%, respectively. Authors [50]

proposed a network intrusion detection algorithm that combined hybrid sampling with the deep hierarchical network where SMOTE was used to create the balanced dataset. The CNN-based Bi-LSTM hybrid technique was used to detect the anomalies on the NSL-KDD and UNSW-NB15 datasets and found the highest accuracy of 83.58% and 77.16%, respectively. The authors [51] proposed a Bi-directional GAN-based approach to the NSL-KDD and CIC-DDoS2019 datasets. The bidirectional GAN model works perfectly on the imbalance NSL-KDD dataset resulting in an accuracy of 91.12% and an f1 score of 92.68%.

The authors used the GAN algorithm to improve the performance of the imbalanced NSL-KDD data. Authors [52] proposed a novel solution based on ACGAN and ACGAN-SVM to solve the data imbalance problem using generative adversarial networks to synthesize the attack traffic for IDS. The synthesized attacks are mixed with the original data to form the augmented dataset. The authors performed experiments on the NSL-KDD, UNSW-NB15, CICIDS2017, and RAWDATA datasets. Among the SVM, DT, and RF models, DT provides a higher F1-score of 92% on the NSL-KDD augmented dataset. During this work [53], the Authors used a Heterogeneous Ensemble Assisted Machine Learning Model for Binary and Multi-Class Network Intrusion Detection to overcome the data imbalance problem on KDD99, NSL-KDD, and UNSW-NB15 datasets. The model provides the 94.5% true positive rate and 96.2% AUC on the NSL-KDD dataset. Authors [54] concluded from the experimental results that the machine learning classifier's performance improved when the number of target classes decreased. Authors examined this concept on traditional machine learning models, including NB, J48, RF, BayesinNet, Bagging, and Adaboost on three NIDS datasets: UNSW-NB15, CIC-IDS2017_Thrusday, and KDD99.

Authors [55] studied the Recurrent Neural Network-based IDS model's performance in binary and multiclass classification. The number of neurons and different learning rates influences the proposed model's performance on the NSL-KDD dataset. The experimental results show that RNN-IDS is suitable for modeling a classification model with high accuracy. Its performance is superior to traditional machine learning (J48, artificial neural network, random forest, and support vector machine) classification methods in binary and multiclass classification. In this paper [56], authors propose a Convolutional Autoencoder-based (CAE) network anomaly detection method and found a detection accuracy of 96.87% on the NSL-KDD dataset. The CAE method was used to reduce and select the more relevant features for the anomaly detection algorithm.

In this paper [57], authors explored the effectiveness of various Autoencoders in detecting network intrusions. The authors compared the performance of 4 different autoencoders, including Sparse Autoencoders, Undercomplete Deep Autoencoders, and Denoising Autoencoder, on the NLS-KDD dataset and achieved an accuracy of 89.34% by using a Sparse Deep Denoising Autoencoder. Authors [58] proposed a 5-layer autoencoder (AE)-based model better suited for network anomaly detection. The optimal model architectures are better equipped for feature learning and dimension reduction to produce better detection accuracy and f1-score by achieving the detection accuracy and f1-score at 90.61% and 92.26%, respectively, on the NSL-KDD dataset. The authors utilized the reconstruction error function to decide whether a network traffic sample is normal or abnormal.

In this paper [59], authors implemented a network intrusion detection method combining CNN and Bi-LSTM network on the KDD99 dataset. The authors studied the

effect of hidden layers, nodes, and the number of iterations to improve anomaly detection accuracy, where the accuracy of KNN, J48, Deep Forest, Naïve Bayes, Random Forest, and CNN-based Bi-LSTM. The CNN-based Bi-LSTM provides the highest detection accuracy of 95.4%. Authors [60] compared the single-layer and multilayer LSTM (4 layers) for weather forecasting on the weather dataset collected by Weather Underground at Hang Nadim Indonesia Airport with the highest validation accuracy of 80.60%. The different numbers of nodes on four hidden layers were used 200, 100, 90, and 50, and the data split ratio taken is 30 % test data for 500 epochs. The Authors [61] implemented the deep learning model based on Bi-directional LSTM on KDDCUP-99 and UNSW-NB15 datasets with outstanding results with 99% accuracy for both KDDCUP-99 and UNSW-NB15 datasets. Most existing models cannot efficiently detect rare attack types, especially User-to-Root (U2R) and Remote-to-Local (R2L) attacks. These two attacks often have lower detection accuracy than other kinds of attacks. Authors in [62] proposed a Bidirectional Long-Short-Term-Memory (Bi-LSTM) based intrusion detection system to handle the aforementioned challenges on the NSL-KDD dataset. This Bi-LSTM model provided an accuracy of 94.26% for binary classification.

The impact of batch size on the performance of CNN and the impact of learning rates were studied for image classification, specifically for medical images [63]. According to their findings, a larger batch size typically does not result in high accuracy, and both the learning rate and the optimizer employed will have a big impact. The network will train more effectively, particularly during fine-tuning, if the learning rate and batch size are reduced.

Various methods were implemented to overcome the data imbalance problem, including data augmentation on [49], SMOTE on [50], GAN technology on [51] [52], Heterogeneous ensemble assisted on [53], reducing the target class combining the smaller class in another new class on [54]. Huge numbers of research works related to network anomaly detection are examined in deep learning, including RNNIDS in [55], CAE in [56], Autoencoder in [57], multilayer Autoencoder in [58], CNN Bi-LSTM hybrid method in [59], and Bi-LSTM in [61] [62].

Authors [62] and [61] do not mention the data preprocessing, train-test data ratio, and how those Bi-LSTM hyperparameters are adopted during their experiments. The authors [60] found Bi-LSTM for weather forecasting without referencing the values of the hyperparameters in their experiments. The authors [55] did not analyze the number of epochs and did not mention the percentages of the split ratio for the KDDTrain+ dataset. Most of the above research works are focused on increasing the model accuracy of either traditional or deep machine learning models. The research on selecting the hyperparameters in deep learning-based models, training testing data ratio, and architecture of deep neural networks are not focused on. Some researchers do not mention how those values are adopted in their research works. Hence, the research focused on improving those limitations on network anomaly detection systems by experimenting with the NSL-KDD dataset.

The main contributions of this research work are:

1) Investigating the effect of optimizers, batch size, the number of epochs Vs performance of the Bi-LSTM.
2) Investing in the train and test split ratio to improve the network anomaly detection accuracy on the NSL- KDD.

3) Investing the number of layers and memory elements to improve the Bi-LSTM on the NSL-KDD dataset.
4) This research presents the development and implementation of network anomaly detection using a Bi- LSTM-based RNN model that can detect anomalies in a network with a high accuracy of 98.52%.

III. SYSTEM MODEL

The overall proposed model encompasses the following steps.

Step-1 Data Collection and Modelling

Step-2 Data Pre-procession

Step-3 Prepare the training and testing dataset

Step-4 Train and Test the Bi-LSTM Model

Step-5 Model Evaluation and Anomaly Detection

Step-6 Model Compare and Decision

The overall implementation schematic of the Bidirectional LSTM-based model is given in Fig. 3.2. A detailed discussion of the above-stated methods is provided in the subsequent sections.

A. Data Collection and Modelling

In this research, we used the KDDTrain+ dataset, one of the datasets available on NSL-KDD. This dataset contains the full NSL-KDD train set, including attack-type labels and difficulty level. It has 41 features with five distinct attack classes, Normal, DoS, Probe, R2L, and U2R. NSL-KDD [64] is an improved version of the KDD99 network intrusion dataset, does not include redundant records in the train set, and has no duplicate records in the test sets. The KDDTrain+ dataset contains 125973 records and 41 features. This dataset

is balanced because 53.46% of records are normal, and 46.54% are abnormal. We selected this dataset because the normal and abnormal records contained the subset of the dataset is balanced.

B. Data Pre-processing

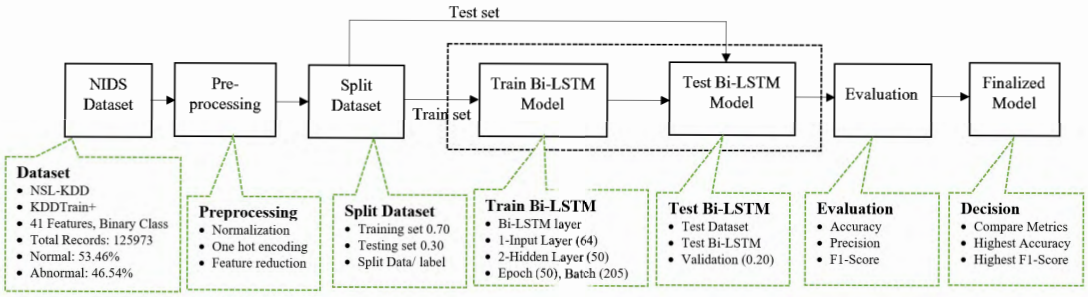

Fig. 3. 2. Block diagram of Bi-LSTM model.

The KDDTrain+ dataset contains 125973 records and 41 features. During the data pre-processing, the class label is assigned 1 for normal and 0 for abnormal records; hence the dataset becomes the binary class dataset. Then, three categorical features: 'protocol_type,' 'service,' and 'flag,' are converted into numeric features using dummy one hot encoding. The standard scalar method is used to normalize the dataset. For the feature reduction, attributes with more than 0.5 correlation with encoded attack label attribute are only preserved.

C. Prepare the Training and Testing Dataset

The train-test split approach measures how well machine learning algorithms perform when used to make predictions from data that was not used to train the model. Since the dataset we pre-processed is only one set of data, the two set of datasets to implement the machine learning algorithms. The train test split ratio does not have rules for the researcher to follow, but the common slit ratio is train 80% and test 20%, train 60% and test 40%, train

70% and test 30%, train 75% and test 25%. We performed the experiment to choose the split ratio where our Bi-LSTM model provides the best result is 70% train and 30% test dataset.

D. Bi-LSTM Model

A recurrent neural network (RNN) consists of feedback loops that process the sequences of data patterns and predict outcomes. Those loops allow the data to be shared with available nodes and predictions according to the collected information called memory. RNN has been used to solve machine learning problems such as speech recognition, language processing, and image classification. LSTM addresses the problem of the vanishing gradients of RNN. LSTM architecture consists of the memory block and three multiplicative units- the input, output, and forget gates which are analogous to write, read and reset operations for the cells. The LSTM memory cells can store and access data for extended periods because of the multiplicative gates, which prevents the vanishing gradient problem.

Conventional RNNs have the limitation that they can only use the previous context. Bidirectional RNN overcomes those problems by processing the data in both directions with two hidden layers, then feeding forwards to the same output layer. Generally, in a normal LSTM network, the output is taken directly. In the case of a bidirectional LSTM network, the output of the forward and backward layers at each stage is given to the activation layer. This output contains information on past and future patterns or data. The bi-directional LSTM predicts or tags the sequence of each element by using finite sequences in the context of previous and subsequent items. This results from two LSTMs processed in series, one from right to left and the other from left to right.

E. Model Evaluation and Anomaly Detection

Different experiments are performed to evaluate the Bi-LSTM model. Machine learning (ML) or deep learning (DL) model does not provide consistency in performance. Hence the model hyper-parameters need to be examined to obtain better performance. The determination of optimizer, the number of epochs, batch size, and train-test split ratio are determined by comparing the accuracy and F1-score of the Bi-LSTM model. Finally, the Bi-LSTM model performance parameters are compared with the previously published research results to evaluate our Bi-LSTM model's performance.

F. Model Comparision and Decision Making

Different sets of experiments to determine the values of the hyperparameters for the best result. The determination of optimizer, the number of epochs, batch size, and train-test split ratio are determined by comparing the accuracy and F1-score of the Bi-LSTM model. Finally, the Bi-LSTM model performance parameters are compared with the previously published research results to evaluate our Bi-LSTM model's performance. The performance metrics are recorded and compared for NSL-KDD binary NIDS datasets regarding f1-score and accuracy.

IV. RESULTS AND DISCUSSIONS

The experiments were adapted on a 64-bit Windows 10 machine with 16G RAM and an i7-1.99GHz processor. The versions of python, Keras, and TensorFlow used during this research work were 3.7.13, 2.6.0, and 2.9.1, respectively. The determination of training and testing data ratio, epochs, batch size, and selection of optimizer for the Bi-LSTM model was examined in the different experiments, which are explained below.

A. Experiment-1 Optimizers Vs Bi-LSTM Model Accuracy

In this experiment, the Bi-LSTM model experimented with the NSL-KDD dataset, whose specifications are given in the previous sections. The right optimizer is necessary for the model to improve training speed and performance. The selection of an optimizer is very important because it helps the ML /DL model to get results faster. TensorFlow supports nine optimizer classes, including Adadelta, Adagrad, Adam, Adamax, Ftrl, Nadam, RMSprop, SGD, and gradient descent were compared. During this experiment, Bi-LSTM hyperparameters were chosen randomly, which are shown in Table 3.1 below. The Bi-LSTM model was created using 64 units, two bidirectional LSTM hidden layers with 50 units, and one output-dense layer. Each layer in Bi-LSTM used a relu activation function and a 20% dropout rate.

TABLE 3. 1
OPTIMIZER VS ACCURACY FOR BI-LSTM

Test size = 50%, epochs = 105, batch Size = 200			
SN	**Optimizer**	**Accuracy %**	**F1-Score %**
1	**Nadam**	**98.35**	**98.47**
2	Adam	98.33	98.44
3	RMSprop	98.28	98.39
4	Adamax	98.07	98.21
5	SGD	96.79	97.03
6	Adagrad	91.65	92.49
7	Ftrl	53.36	69.59

Observing the above results (see Table 3.1), it can easily be found that the Nadam optimizer is the winning optimizer with the highest accuracy of 98.35% and the highest f1-score of 98.47%. Nadam is an improved version of the Adam algorithm that integrates Nesterov momentum, improving the optimization algorithm's performance.

B. Experiment-2 Train Test Ratio Vs. Accuracy

The train-test split ratio and Bi-LSTM model accuracy were studied in this experiment. Data splitting is crucial in data science, especially when building models from data. The train-test split approach is used to quantify how well machine learning algorithms perform when used to predict outcomes from data that was not used to train the model. After the training is completed, the testing data set is utilized. There is no set guideline for how the data should be split on training and test data from the given data set. The test split ratio is examined to obtain better network anomaly detection using Nadam optimizer on the NSL-KDD binary dataset.

TABLE 3. 2
TRAIN TEST RATIO VS. ACCURACY ON BI-LSTM

Optimizer = Nadam, epochs = 105, batch Size = 200			
SN	**Test Data size %**	**Accuracy %**	**F1-Score %**
1	**30**	**98.48**	**98.57**
2	25	98.47	98.57
3	50	98.39	98.5
4	40	98.35	98.46
5	20	98.33	98.44
6	60	98.28	98.4
7	10	98.17	98.29
8	70	98.15	98.29
9	80	98.15	98.29
10	90	97.98	98.13

This experiment provides the train-test ratio for the highest network anomaly detection for the Bi-LSTM model on the NSL-KDD dataset. The performance metrics are recorded in Table 3.2, where the test split of 30% achieved the proposed Bi-LSTM model with the highest accuracy and f1-score of 98.48% and 98.57%, respectively.

C. Experiment-3 Batch Size Vs. Bi-LSTM Accuracy

The effect of the batch sizes on the Bi-LSTM accuracy and the training time was studied during this experiment. This experiment aims to find the optimal batch size for the best model performance.

TABLE 3. 3
BATCH SIZE VS. ACCURACY ON BI-LSTM

Optimizer = Nadam, epochs = 105, test_size= 0.30				
SN	**Batch Size**	**Accuracy %**	**F1-Score %**	**Prgm Exe time (sec)**
1	**50**	**98.48**	**98.58**	2127.2346
2	100	98.46	98.56	1228.779
3	15	98.45	98.56	5671.738
4	200	98.45	98.55	796.8976
5	300	98.45	98.55	553.4444
6	150	98.42	98.52	858.07
7	450	98.41	98.51	454.989
8	350	98.4	98.51	527.1532
9	400	98.38	98.48	460.8835
10	500	98.36	98.47	514.7698
11	250	98.35	98.46	616.4657

The smaller batch size introduces small amounts of data samples and takes longer to train the Bi-LSTM model compared to the larger batch size. Model accuracy, F1-score, is shown in Table 3.3. The experimental result shows that the batch size of 50 during this Bi-LSTM model for the NSL-KDD dataset produces the best results in terms of accuracy and f1-score. Larger batch sizes take less time to train but are less accurate, which is an important trade-off for this Bi-LSTM model.

D. Experiment-4 Epochs Vs. Bi-LSTM Accuracy

The number of times the learning algorithm will go over the complete training dataset is determined by the hyperparameter known as the epoch. The number of epochs can be any integer value that lies between 1 to infinity. Traditionally, the ML/ DL model uses large values of epochs.

TABLE 3. 4
EPOCHS VS. BI-LSTM MODEL ACCURACY

Optimizer = Nadam, batch_size = 50, test_size= 0.30				
SN	**Epochs**	**Accuracy %**	**F1-Score %**	**Prgm Exe time (sec)**
1	**205**	**98.52**	**98.62**	4103.7667
2	100	98.48	98.58	1878.8025
3	125	98.48	98.58	2470.6198
4	150	98.48	98.58	2934.2485
5	175	98.48	98.58	3965.207
6	75	98.46	98.56	1465.5138
7	50	98.38	98.48	942.1289
8	45	98.37	98.47	1002.0923
9	35	98.35	98.46	761.2784
10	25	98.3	98.41	527.5244
11	15	98.13	98.25	322.5288
12	5	97.9	98.03	127.0577

This experiment aims to determine the epochs where the Bi-LSTM model provides the highest accuracy. During this experiment, Bi-LSTM hyperparameters were chosen randomly same as in the previous experiment. The larger epochs take a longer time to train the model. We chose epochs ranging from 5 to 205 with some intervals; the accuracy and f1-score are higher for 205 epochs. The training time for Bi-LSTM is increased for a large value of epoch. During this experiment, a batch size of 205 improves the Bi-LSTM model's accuracy of 98.52% in detecting network anomalies.

E. Experiment-5 Bi-LSTM layers parameters Vs. Accuracy

We investigated the optimizer, epochs, batch size, and train test data split ratio from the above experiments A-D and found that the value of the hyperparameter: Nadam optimizer, 205 epochs, 50 batch size, 30% test data, and 70% train data generate the best performance which is measured using performance evaluation metrics. During this experiment, we examined the combination of the numbers of units for the multilayer Bi-LSTM model. The output layer provides the probability of selecting either a normal or

abnormal class, so the softmax activation function works best for binary class classification problems.

TABLE 3. 5
BI-LSTM LAYERS PARAMETERS VS. ACCURACY

Optimizer = Nadam, batch_size = 50, test_size= 0.30 [Units (activation fn)]				
SN	**Input Layer**	**Hidden Layer 1**	**Hidden Layer 2**	**Accuracy**
1	64 (relu)	50 (relu)	50 (relu)	98.52
2	80 (relu)	64 (relu)	64 (relu)	98.48
3	49 (sigmoid)	128 (Sigmoid)	128 (sigmoid)	98.18
4	16 (selu)	16 (selu)	16 (selu)	97.97
5	16 (relu)	16 (relu)	16 (relu)	97.93
6	4 (relu)	4 (relu)	4 (relu)	97.55
7	8 (relu)	8 (relu)	8 (relu)	97.48
8	4 (sigmoid)	4 (sigmoid)	4 (sigmoid)	97.05

The number of combinations of Bi-LSTM units and activation functions was used in input and hidden layers during this experiment; some of the experiment results are included in Table 3.4. The experimental result shows that the 64 Bi-LSTM units in the input layer and 50 Bi-LSTM units in both hidden layers produce the highest accuracy of 98.52% during network anomaly detection.

V. CONCLUSION

Comparing the result with existing research [61] for Bi-LSTM, the model produces a higher accuracy of 98.52%, which is greater than 94.26%. The values of Bi-LSTM model hyperparameters, including optimizer, epochs, batch size, and the training testing dataset ratio for the multilayer Bi-LSTM neuron architecture (layers, activation function, and memory units) are investigated for the highest detecting accuracy. All the above experimental results show that the Bi-LSTM model with those investigated parameters can effectively improve the detection accuracy and f1-score.

CHAPTER 4. EFFICACY OF CNN-BIDIRECTIONAL LSTM HYBRID MODEL FOR NETWORK-BASED ANOMALY DETECTION

Abstract- With the development of the web and the internet, computer networks have become an important tool to transfer information digitally, which increases the system's threats and vulnerability. Cyber attackers can use the internet and tools to compromise the triad of the CIA (Confidentiality, Integrity, and Availability). Network anomaly detection is challenging while detecting anomalous behavior in a network due to the large-scale data, imbalance nature of attack class, and huge numbers of features in the dataset. Traditional Machine learning methods are not very efficient in solving those problems. Deep learning has proven to be more efficient in detecting network-based anomalies. A Recurrent Neural Network (RNN) model is designed to recognize the sequential data characteristics to predict. We proposed a convolutional neural network with bidirectional long-short memory (CNN Bi-LSTM) model to analyze the hyperparameters, including optimizers (Nadam, Adam, RMSprop, Adamax, SGD, Adagrad, Ftrl), epochs, batch size, learning rate, and neural network model architecture of CNN-BLSTM algorithms. Those analyzed hyperparameters provide the highest anomaly detection accuracy of 98.27% and 99.87% on the NSL-KDD and UNSW-NB15, respectively. Performance assessment regarding the accuracy and F1-score revealed that the proposed CNN Bi-LSTM anomaly detection model exhibited better performance than the other existing anomaly detection methods.

Keywords— Bi-LSTM, CNN, deep learning, LSTM, machine learning, network intrusion detection system, NSL-KDD, UNSW-NB15

I. INTRODUCTION

As technology develops rapidly, the method of transmission of information from source to destination has evolved through the wired, wireless, or guided network. The development of network technology plays a vital role in people's daily activities. Any system is considered secure if the three computer security principles of confidentiality, integrity, and availability (CIA) are properly met. Hence information security is securing information from an unauthorized agent, preventing access, use, disclosure, modification, recording, or data destruction.

A firewall and antivirus software cannot completely protect the traditional network. The antivirus and firewall detect those activities already defined as anomalous and set the rule to block those activities by the expert. Outliers and anomalies are sometimes used interchangeably in anomaly detection. Anomaly detection has abundant applications, including business, network intrusion detection, health monitoring systems, credit card fraud detection, and fault detection in critical information systems. Anomaly detection is important in cyber security for solid protection against cyber adversaries. There must be secure network resources against cyber threats to protect the system.

Anomalies are classified as point, contextual, and collective according to the output from the detection method used [48]. Point anomaly occurs when a certain behavior deviates from the regular pattern. Contextual anomalies are strange patterns in a particular context that always differ from many normal behaviors. The collective anomaly occurs when a group of similar instances acts anomalously competed with the dataset of normal activities.

There are two categories of intrusion detection methods: signature-based intrusion detection systems (SIDS) and anomaly-based intrusion detection systems (AIDS). Anomaly

detection systems are classified into two categories based on the sources: network-based and host-based intrusion detection systems. Anomaly detection techniques utilize labels to identify whether the data is normal or anomalous. There are three different anomaly detection techniques such as supervised, unsupervised, and semi-supervised anomaly detection methods. AIDS overcomes the SIDS's drawbacks by modeling normal behaviors using machine learning (ML), statistical-based, or knowledge-based methods. Anomaly-based detection can also produce false results caused by changes in user habits.

Most traditional machine learning algorithms are shallow learning methods emphasizing feature engineering suited for small datasets. Feature engineering requires time and domain expertise to generate the features and remove those irrelevant features from the anomaly detection model. The anomaly detection performance depends on how the feature engineering is implemented and the data preprocessed carried out. The traditional ML methods are simple, have low resource consumption, and perform poorly on computer vision, natural language processing, and image translations.

CNN is mostly used in image datasets where the lower layer’s neurons reduce the network’s features, usually identifying important small-scale features, such as boundaries, corners, and intensity differences. Then in higher layers, the network combines the lower-level features to form more complex features such as simple shapes, forms, and partial objects. And on the final layer, the network combines the lower features to produce the output or classification results.

An LSTM works differently than a CNN because an LSTM is designed to retain long-range information so that the information is remembered and not lost in a long

sequence. Bi-LSTM adds one more LSTM layer, reversing the information flow direction and overcoming the vanishing gradient problems.

The deep learning method overcomes the problems in traditional ML. The performance of the deep learning-based anomaly detection algorithm depends on neural network architecture, number of hidden layers, types of activation function, number of samples (batch size), and epochs during DL model training and testing. Selecting those hyperparameters and architecture of neural networks in deep neural networks is vital in increasing the detection accuracy of network anomaly detection systems.

II. RELATED WORK

Due to the development of information and technology, many end terminals are connected to the internet and network. The most terminal connected to the internet are smart, and they generate a vast amount of data called big data. Machine learning and deep learning algorithms process the data and make predictions from observations and data that generate valuable insights. The volume of big data is growing daily, so the traditional machine learning algorithm cannot be performed well and needs intensive feature engineering tasks. Deep learning greatly improves detection performance. Still, the nature of the dataset, feature engineering, the hyperparameters on deep neural networks, and neural network architecture plays a vital role in detecting the anomaly in network intrusion detection systems.

Traditional ML depends heavily on feature engineering, which is often time-consuming, complex, and impractical during real-time applications. Authors [65] purposed CNN and RNN-based payload classification approach to detect attacks and achieved an accuracy of 99.36% and 99.98%, respectively, on the DARPA98 dataset. Authors [66]

proposed the CNN with Gated Recurrent Unit (GRU) model to address the class imbalance problem by adapting a hybrid sampling algorithm combining Adaptive Synthetic Sampling (ADASYN) and Repeated Edited nearest neighbors (RENN). Random forest and Pearson correlation analysis were used to solve the feature redundancy problem. Their CNN-GRU model outperformed with an accuracy of 86.25%, 99.69%, and 99.65% on UNSW_NB15, NSL-KDD, and CIC-IDS2017 datasets, respectively.

Authors [49] proposed that the deep learning-based network intrusion detection model used adaptive synthetic sampling (ADASYN) to balance the dataset. The autoencoder is used to reduce dimensionality on NSL-KDD. The CNN-BLSTM-based deep learning model provided the highest accuracy and F1 score of 90.73% and 89.65%, respectively. Authors [67] federal transfer learning and convolutional neural networks to solve the problem that arises from data imbalance and different data distribution from the different information sources. The model provided average model accuracy of 86.85% on the UNSW-NB15 multiclass network dataset. Authors [53] used a Heterogeneous Ensemble Assisted Machine Learning Model for Binary and Multi-Class Network Intrusion Detection to overcome the data imbalance problem on KDD99, NSL-KDD, and UNSW-NB15 datasets. The model provides the 94.5% true positive rate and 96.2% AUC on the NSL-KDD dataset. In [54], the Authors concluded from the experimental results that the machine learning classifier's performance improved when the number of target classes decreased. Authors examined this concept on traditional machine learning models, including NB, J48, RF, BayesinNet, Bagging, and Adaboost on three NIDS datasets: UNSW-NB15, CIC-IDS2017_Thrusday, and KDD99.

Authors [68] proposed the method to achieve a successful classification with low computational cost by grouping attributes according to the conditions on which they are collected and creating the cluster attributes for each group with K-means with an accuracy of 98.84% on the KDD99 dataset. The detection accuracy for U2R is very low, 21.92%, which reduces the overall model performance. The authors [69] implemented the hybrid approach combining the CNN and LSTM to improve the anomaly classification accuracy of 98.1% and 96.7% on NSL-KDD and CICIDS2017 datasets, respectively. Authors [70] proposed the hybrid model combining CNN and LSTM to improve the intrusion detection capabilities of advanced metering infrastructure (AMI) utilizing the cross-layer features fusion. The model produced the highest accuracy of 99.95% on KDD Cup99 and 99.79% on the NSL-KDD dataset, having low U2R detection capabilities. Authors [71] implemented the hybrid network of CNN and LSTM to improve intrusion detection to extra network traffic data's spatial and temporal features.

Authors [72] in this paper implemented the method based on the mean control of the CNN-BLSTM algorithm to overcome the traditional data preprocessing and unbalanced numerical distribution on the NSL-KDD dataset, providing the highest accuracy of 99.10%. Still, accuracy for the fewer data class shows poorly. Authors [73] proposed a DL model combining with CNN and Bidirectional LSTM to incorporate the learning of spatial and temporal features of the data on the accuracy of 93.84% and 99.30% and binary class UNSW-NB15 and NSL-KDD datasets, respectively. Authors [74] used CNN Bi-LSTM algorithms on multiclass NSL-KDD dataset and obtained an accuracy of 96.3% where one-hot encoding and min-max normalization are used during data preprocessing. Authors [59] implemented the CNN Bi-LSTM algorithm on preprocessed and obtained an accuracy of

95.4% on the NSL-KDD dataset. The C5.0 decision tree model is combined with the CNN Bi-LSTM model to skip the design feature selection and directly learn the model to represent features of high dimensional data. The Authors [61] implemented the deep learning model based on Bi-directional LSTM on KDDCUP-99 and UNSW-NB15 datasets with outstanding results with 99% accuracy for both KDDCUP-99 and UNSW-NB15 datasets. Most existing models cannot efficiently detect rare attack types, especially User-to-Root (U2R) and Remote-to-Local (R2L) attacks. These two attacks often have lower detection accuracy than other kinds of attacks. Authors in [62] proposed a Bi-LSTM-based intrusion detection system to handle the aforementioned challenges on the NSL-KDD dataset. This Bi-LSTM model provided an accuracy of 94.26% for binary classification. The authors [51] proposed a Bi-directional GAN-based approach to the NSL-KDD and CIC-DDoS2019 datasets. The bidirectional GAN model works perfectly on the imbalance NSL-KDD dataset resulting in an accuracy of 91.12% and an f1 score of 92.68%.

The deep learning-based model in [65], [66] overcome traditional ML problems to detect the anomaly. Data imbalance problems are addressed [49], [67], [53], and [54] . Feature engineering is the most important factor in improving the accuracy of the ML/DL model. Huge numbers of research have been done related to feature engineering, grouping attributes in [68], [69], [70], [71]. A Bi-LSTM combines two separate LSTMs to permit running input in two directions from the past to the future and from the future to the past to improve the traditional LSTM. Bi-LSTM was implemented in [72], [73], [74], [59], [61], [62], [51] to improve the model anomaly detection accuracy.

Most of the above research works focus on increasing the accuracy of traditional or deep machine learning models, working for feature engineering and data imbalance. The

research on selecting the hyperparameters in deep learning-based models, training testing data ratio, and architecture of deep neural networks are not focused on. Some researchers do not mention how those values are adopted in their research works. Hence, this research focused on improving those limitations on network anomaly detection systems by experimenting with the NSL-KDD and UNSW-NB15 datasets.

The main contributions of this research work are:

1) Investigating the effect of CNN Bi-LSTM architecture Vs. performance of CNN Bi-LSTM.
2) Investigating model performance Vs. Hyperparameters on both NIDS datasets, i.e., NSL-KDD and UNSW-NB15.
3) Investing the number of layers and memory elements to improve the CNN Bi-LSTM.
4) This research presents the development and implementation of network anomaly detection using a CNN Bi-LSTM model that can detect anomalies with high accuracy of 98.27 % and 99.87% on NSL-KDD and UNSW-NB15, respectively.

The remainder of the paper is as follows. Section II describes the system model of our proposed CNN Bi-LSTM approach. Section III illustrates the results and discussion, while Section IV concludes this research work.

III. SYSTEM MODEL

The overall proposed model encompasses the following steps.

Step-1 Data Collection

Step-2 Data Pre-processing

Step-3 Prepare the training and testing dataset

Step-4 Train and Test CNN Bi-LSTM model

Step-5 Model Evaluation and Anomaly Detection

Step-6 Model Compare and Decision

The overall implementation schematic of the CNN bidirectional LSTM-based model is shown in Fig. 4.1. A detailed discussion of the above-stated methods is provided in the subsequent sections. In Fig. 4.2., the detailed architecture of neural networks and CNN and Bi-LSTM layers components are clearly shown.

A. Data Collection and Modelling

This research used two datasets, NSL-KDD KDDTrain [64] + and UNSW-NB15, where The KDDTrain+ dataset contains the full NSL-KDD train set, including attack-type labels and difficulty level. It has 41 features with five distinct attack classes, Normal, DoS, Probe, R2L, and U2R. Typically, these features are classified into various groups, such as basic, content, and time-based features. NSL-KDD is an improved version of the KDD99 network intrusion dataset, does not include redundant records in the train set, and has no duplicate records in the test sets. The KDDTrain+ dataset contains 125973 records and 41 features. This dataset is balanced because 53.46% of records are normal, and 46.54% are abnormal.

The Australian Centre for Cyber Security (ACCS) cybersecurity research team created the UNSW-NB15 dataset [75] to solve issues with the KDD99 dataset. The data used in this research comprises 42 features. This dataset consists of various attacks, including Analysis, Backdoor, DoS, Exploit, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms counts of 2677, 2329, 16353, 44525, 24246, 58871, 13987, 1511, and 174, respectively. The normal traffic of 93000 data makes the total data 257673.

B. Data Pre-processing

During the KDDTrain+ data preprocessing, the class label is assigned 1 for normal and 0 for abnormal records; hence the dataset becomes the binary class dataset. Then, three categorical features: 'protocol_type,' 'service,' and 'flag,' are converted into numeric features using dummy one hot encoding. The standard scalar method is used to normalize the dataset. For the feature reduction, attributes with more than 0.5 correlation with encoded attack label attributes are only preserved, resulting in 93 features on the final dataset.

UNSW-NB15 data sets consist of test and training separate files. Both contain 45 features, including attack categories and labels. The same methods are used to preprocess both test and training files. Dummy one hot encoding is used for categorical features (proto, service, state), and the standard scalar method is used to normalize the numerical features before combining them. The empty columns are inserted in the location where the features are missed after one hot encoding. All attack categories are grouped into a single attack category to create the binary dataset. After preprocessing, the training and test data sizes become (82332, 199) and (175341. 199), respectively.

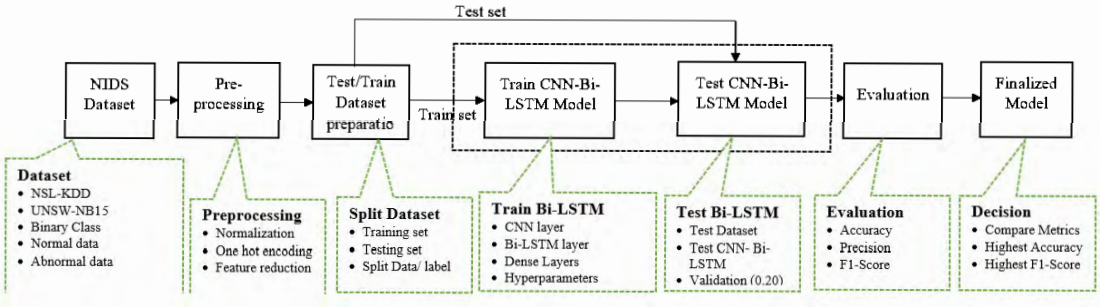

Fig. 4. 1 Block diagram of CNN-BLSTM model.

C. Prepare the Training and Testing Dataset

The train-test split approach measures how well machine learning algorithms perform when used to make predictions from data that was not used to train the model. We choose the 70:30 split ratio where our CNN Bi-LSTM model for KDDTrain+ dataset with 70% train and 30% test datasets. There are two separate files chosen in the case of UNSW-NB15, one for training and another for testing the model. The details about the number of training and testing data are explained in the data preprocessing section above.

D. Bi-LSTM Model

Convolutional Neural Network (CNN) are deep neural networks that can recognize and classify using the image format. CNN used the convolutional operation to identify the various features of the images then pooling layers extracts the features and a fully connected layer that utilizes the output from the previous layer to classify. Convolutional layers and pooling layers are used for feature extraction whereas the last fully connected dense layer is used for classification purposes.

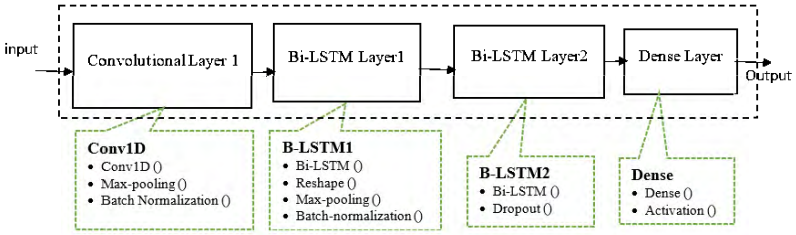

Fig. 4. 2. CNN Bi-LSTM model architecture.

A recurrent neural network (RNN) consists of feedback loops that process the sequences of data patterns and predict outcomes. RNN consists of memory to store the previous and future state information. RNN has been used to solve machine learning problems such as speech recognition, language processing, and image classification. LSTM addresses the problem of the vanishing gradients of RNN. LSTM architecture consists of the memory block and three multiplicative units- the input, output, and forget gates which are analogous to write, read and reset operations for the cells. The LSTM memory cells can store and access data for extended periods because of the multiplicative gates, which prevents the vanishing gradient problem. A bidirectional RNN often combines two separate RNNs to permit running input in two directions: from the past to the future and from the future to the past. The forward and backward LSTM networks comprise the two LSTM networks that comprise the Bi-LSTM. The goal of the forward LSTM hidden layer is to extract features in the forward direction, and the backward one is to extract features in the backward direction. The bi-directional LSTM predicts or tags the sequence of each element by using finite sequences in the context of previous and subsequent items. This results from two LSTMs processed in series, one from right to left and the other from left to right. The CNN and Bi-LSTM model consists of several layers with hyperparameters. The CNN Bi-LSTM architecture is shown in Fig. 4.2.

E. Model Evaluation and Anomaly Detection

Machine learning (ML) or deep learning (DL) model does not provide consistency in performance. Hence the model hyper-parameters need to be examined to obtain better performance. The determination of optimizer, the number of epochs, batch size, dropout, and learning rate are determined by comparing the accuracy and F1-score of the Bi-LSTM model. Finally, the CNN Bi-LSTM model performance parameters are compared with the previously published research results to evaluate our Bi-LSTM model's performance.

F. Model Comparision and Decision Making

Different sets of experiments to determine the values of the hyperparameters for the best result. The determination of optimizer, the number of epochs, batch size, and train-test split ratio are determined by comparing the accuracy and F1-score of the Bi-LSTM model. Finally, the Bi-LSTM model performance parameters are compared with the previously published research results to evaluate our Bi-LSTM model's performance. The performance metrics for NSL-KDD and UNSW-NB15 binary NIDS datasets regarding f1-score and accuracy are recorded and compared.

IV. RESULTS AND DISCUSSIONS

The experiment was performed on the Anaconda Navigator Jupyter python platform installed on the central processing unit encompassing a 64-bit Windows 10 machine with 16G RAM and an i7-1.99GHz processor. The versions of python, Keras, and TensorFlow used during this research work were 3.7.13, 2.6.0, and 2.9.1, respectively.

The model architecture shown in Fig. 4.2. consists of 1 convolution layer with 16 units, max-pooling, and batch normalization, Bi-LSTM layer 1 with 50 memory units, reshape, max-pooling, and batch normalization; the Bi-LSTM layer 2 with 100 memory

units and dropout. Finally, the output is taken using a Dense layer with a sigmoid activation function. The model detection accuracy is compared by tuning optimizers, learning rate, epochs, batch size, and dropout rate in the different experiments on NSL-KDD and UNSW-NB15 datasets, which are explained below.

A. Experiment-1 Optimizers Vs. Bi-LSTM Performance

During the training of the CNN Bi-LSTM model, the selection of an optimizer is very important because the helps the ML /DL model to get results faster. Based on the algorithms used by the optimizer, TensorFlow supports nine optimizer classes, including Adadelta, Adagrad, Adam, Adamax, Ftrl, Nadam, RMSprop, SGD, and gradient descent. During the optimizer Vs. Accuracy calculation experiment, the relu activation function, and a 20% dropout rate are used on the model and experimented with seven optimizers, including Nadam, Adam, RMSprop, Adamax, SGD, Adagrad, and Ftrl to find the best optimizer for our model. The performance metrics are recorded in Table 4.1. The results found that the Nadam optimizer is the winning optimizer for NSL-KDD, and adam optimizer provides the highest accuracy for the UNSW-NB15 dataset. Two optimizers perform differently for both NIDS datasets; even the same model architecture is used.

TABLE 4. 1
OPTIMIZERS VS. PERFORMANCE ON CNN-BLSM

Epochs = 10, Batch Size = 256					
SN	**Optimizer**	**ACC_NSL**	**F1_NSL**	**ACC_UN**	**F1_UN**
1	**Nadam**	**98.13**	**98.26**	99.11	99.34
2	**Adam**	98.02	98.16	**99.15**	**99.38**
3	RMSprop	97.87	98.01	97.93	98.46
4	Adamax	97.65	97.78	95.33	96.51
5	SGD	97.74	97.91	99.14	99.37
6	Adagrad	96.98	97.21	94.043	95.62
7	Ftrl	53.47	69.68	0.8099	80.99

B. Experiment-2 Learning Rate Vs. Performance

The same model architecture is used to find the learning rate for better model performance where the optimizers are selected from the previous experiment [4A]. The learning rate determines how the neural network model weights are updated. The learning rates vary to tune the model accuracy, keeping the other hyperparameters unchanged during this experiment. The learning rate Vs. CNN Bi-LSTM model performance is tabulated in Table 4.2. The model provides the highest performance at a learning rate of 0.01 on UNSW-NB15 and a learning rate of 0.0002 on the NSL-KDD dataset. The same learning rate provides different model performances.

TABLE 4. 2
LEARNING RATE VS. PERFORMANCE ON CNN-BLSTM

epochs = 10, batch Size = 256, KDD (Nadam), UNSW-NB15 (adam)					
SN	**LR**	**ACC_NSL**	**F1_NSL**	**ACC_UN**	**F1_UN**
1	**0.01**	97.49	97.67	**99.67**	**99.76**
2	0.001	98.16	98.29	99.54	99.66
3	0.0001	98.06	98.2	95.81	96.85
4	**0.0002**	**98.18**	**98.3**	97.9	98.44
5	0.0003	98.14	98.27	98.44	98.86
6	0.0004	97.97	98.11	99.13	99.35
7	0.0005	98.11	98.25	99.09	99.32

C. Experiment-3 Drop out Vs. Performance

The dropout rate refers to dropping the neurons during the training model to prevent overfitting. The CNN Bi-LSTM model was trained and tested using epochs of 10 batch size 256 for both datasets. Different values of dropout rate are chosen to study the model performance. The model performs better at a dropout rate of 30% on UNSW-NB15, and a 60% dropout rate performs better on the NSL-KDD dataset. The hyperparameter values, dropout rates, and performance are tabulated in Table 4.3. The experiment results show the different drop rates for different datasets even though both data sets are similar.

TABLE 4. 3
DROP OUT VS. PERFORMANCE ON CNN-BLSTM

epochs = 10, batch Size = 256, KDD (Nadam), UNSW-NB15 (adam)					
SN	**DropOut**	**ACC_NSL**	**F1_NSL**	**ACC_UN**	**F1_UN**
1	0.1	98.1	98.24	97.44	98.15
2	0.2	98.02	98.16	98.98	99.25
3	**0.3**	98.16	98.29	**99.87**	**99.9**
4	0.4	98.04	98.17	99.27	99.47
5	0.5	97.93	98.09	99.47	99.61
6	**0.6**	**98.21**	**98.33**	99.81	99.86
7	0.7	98.01	98.15	99.58	99.69
8	0.8	98.04	98.18	98.57	98.94

D. Experiment-4 Batch Size Vs. Performance

Batch size is the number of samples utilized in a single iteration. The smaller batch size introduces small amounts of data samples and takes longer to train the CNN Bi-LSTM model compared to the larger batch size. The batch size is varied, keeping the other hyperparameters fixed, such as epochs of 5, optimizer's learning rate, and dropout rate values assigned on the model to the respective dataset based on the previous experiment's (Experiment 1-3) finding.

TABLE 4. 4
BATCH SIZE VS. PERFORMANCE ON CNN-BLSTM

epochs = 5, KDD (Nadam), UNSW-NB15(adam)					
SN	**batch_size**	**ACC_NSL**	**F1_NSL**	**ACC_UN**	**F1_UN**
1	**32**	97.89	98.04	**99.40**	**99.55**
2	64	97.95	98.10	99.35	99.52
3	**128**	**98.06**	**98.20**	99.33	99.50
4	256	97.64	97.79	96.36	97.26
5	512	97.92	98.08	96.90	97.70

This experimental result in Table 4.4 shows that the combination of hyperparameters in the neural network provides a different performance. During this experiment, the CNN Bi-LSTM model performed better when batch size is 128 for NSL-KDD and 32 for UNSW-NB15 datasets with epochs of 5.

E. Experiment-5 Epochs Vs. Performance

The number of times the learning algorithm will go over the complete training dataset is determined by the hyperparameter known as the epoch, which can be any integer value that lies between 1 to infinity. The model takes a long time to train when we choose smaller epoch values and vice versa. The CNN Bi-LSTM model performance for different values of epochs and assigned the other hyperparameters values found from previous experiments are recorded in Table 4.5.

TABLE 4. 5
EPOCHS VS. PERFORMANCE ON CNN-LSTM

Batch size = 256, KDD (Nadam)			
SN	**Epochs**	**ACC_NSL**	**F1_NSL**
1	2	95.48	95.94
2	10	98.13	98.26
3	25	98.21	98.33
4	50	98.20	98.33
5	**75**	**98.27**	**98.39**
6	100	98.26	98.39

V. CONCLUSION

The literature review shows that the NSL-KDD and UNSW-NB15 have an average model accuracy of 99%, but the smaller attack class such as U2R and R2L, detection is very low. The enemy is the enemy, and every attack is responsible for destroying network machines equally. Hence compare the result with the existing result of 91.12% [51], and 90.83% [49] accuracy for NSL-KDD and 99.70% [61], 82.08% [73] 82.08% for the UNSW-NB15 dataset. The experiment improves accuracy, which is 98.27% on NSL-KDD and 99.87% on UNSW-NB15 binary dataset. The values of CNN Bi-LSTM model hyperparameters, including optimizer, epochs, batch size, the learning rate, and dropout for

the CNN Bi-LSTM neuron architecture, are investigated for the highest detecting accuracy for binary NSL-KDD and UNSW-NB15 dataset.

CHAPTER 5. OPTIMIZING THE PERFORMANCE OF NETWORK ANOMALY DETECTION USING BIDIRECTIONAL LONG SHORT-TERM MEMORY (BI-LSTM) AND OVER-SAMPLING FOR IMBALANCE NETWORK TRAFFIC DATA

Abstract- 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 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. The 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.

Keywords—Bidirectional-LSTM, data imbalance, deep learning, machine learning, network anomaly detection, NSL-KDD, Random Under Sampling (RUS), Random Over Sampling (ROS), sampling, SMOTE.

I. INTRODUCTION

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. Fig. 5.1. provides brief information about the taxonomy of anomaly detection methods [48].

The security of information resources is ensured when the three fundamental principles of computer security—Confidentiality, Integrity, and Availability (CIA)—are

appropriately obeyed [47]. 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 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.

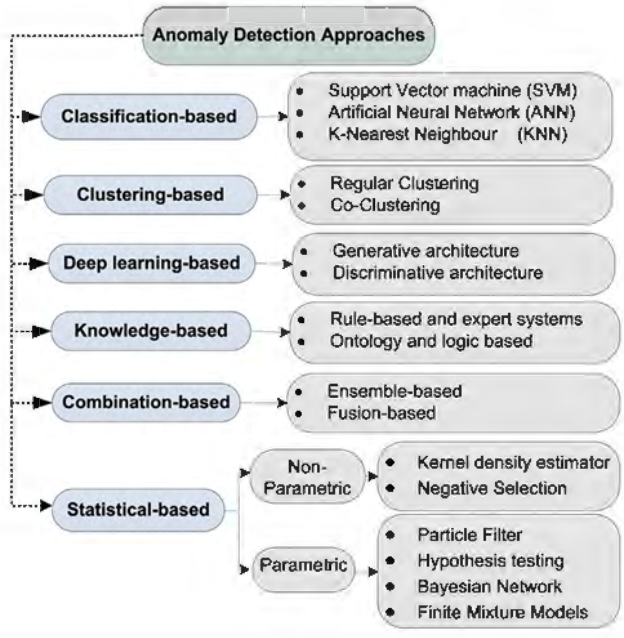

Fig. 5.1. Taxonomy of anomaly detection. [48].

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. The collected data determines the types of anomaly detection whether host-based or network-based anomaly detection.

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 is 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. Fig 5.1 outlines the various approaches to anomaly detection [48].

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.

II. 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 [49]. 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 data set [50]. The SMOTE techniques used to balance the dataset using oversampling method. 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 [51]. 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 [52]. 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 [53], 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 [54] 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 [55], 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 [56], 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 [57], 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 [58], 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 [59], [76], 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 [60], 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 [61] 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 [62] 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 [63] 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 [49], application of SMOTE detailed in [50], the use of GAN technology explored in [51], [52] the assistance

of Heterogeneous ensemble methods investigated in [53], and the reduction of the target class by combining smaller classes into a new category, as discussed in [54]. 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 [55], CAE featured in [56], Autoencoder examined in [57], multilayer AE explored in [58], convolutional neural network combined with bidirectional LSTM hybrid methods presented in [59], [76] and Bi-LSTM discussed in [61], [62].

The researchers in [61] and [62] 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 [60] conducted weather forecasting using Bi-LSTM without specifying the hyperparameter values. In [55], 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.

III. 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 improve the bidirectional LSTM on the NSL-KDD.

The careful choice of machine learning and 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.

IV. 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

Fig. 5.3 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.

A. Data Collection and Modelling

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

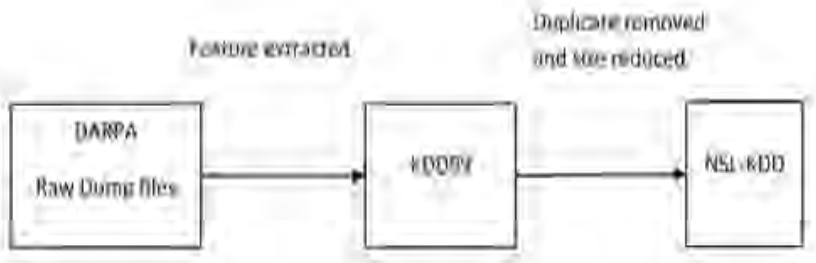

Fig. 5.2. DARPA, KDD99, and NSL-KDD dataset.

The NSL-KDD data is derived from the DARPA KDD99 data, as depicted in Fig. 5.2, 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 [64] 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.

B. 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 the reference [64] 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+_20% is a subset of the training dataset, encompassing 20% of the entire training dataset. It is important to note that the traffic records found in KDDTest-21 and KDDTrain+_20% 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.

Machine 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

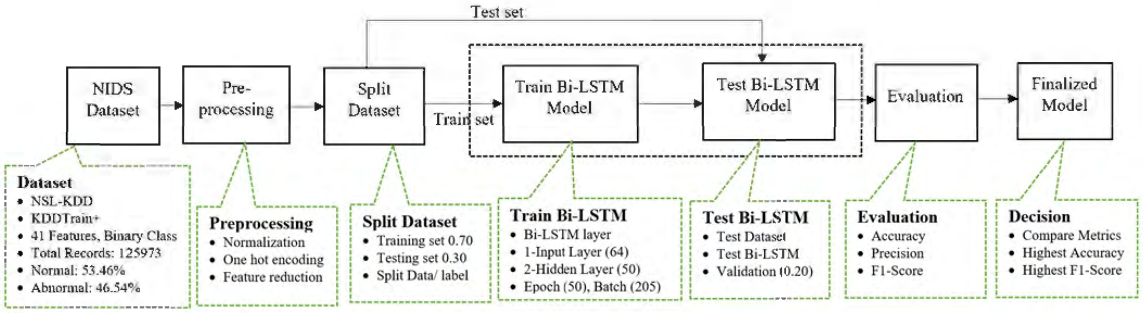

Fig. 5. 3. Bidirectional LSTM model block diagram.

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.

C. 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 the selection. Since the 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.

D. 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.

E. 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 are 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.

F. 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.

V. 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.

A. 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 5. 1
OPTIMIZER VS. ACCURACY ON BI-LSTM

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 5.1. 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%.

Observing the above results (see Table 5.1 and Figure 5.4), 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.

B. 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.

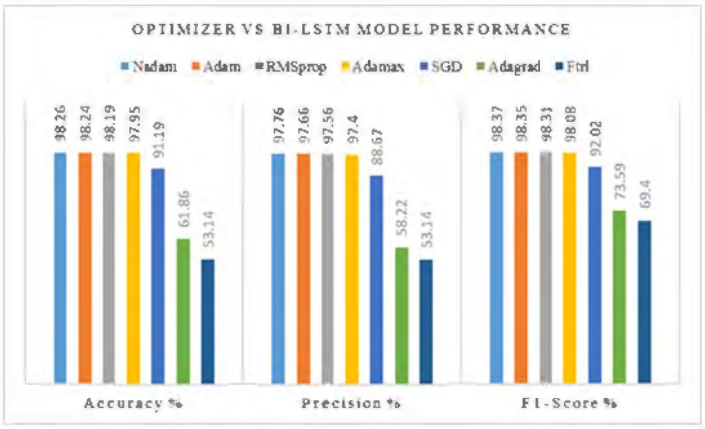

Fig. 5.4. Optimizer vs. accuracy on Bi-LSTM plot.

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 5. 2
TRAIN TEST SPLIT RATIO VS. PERFORMANCE ON BI-LSTM

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

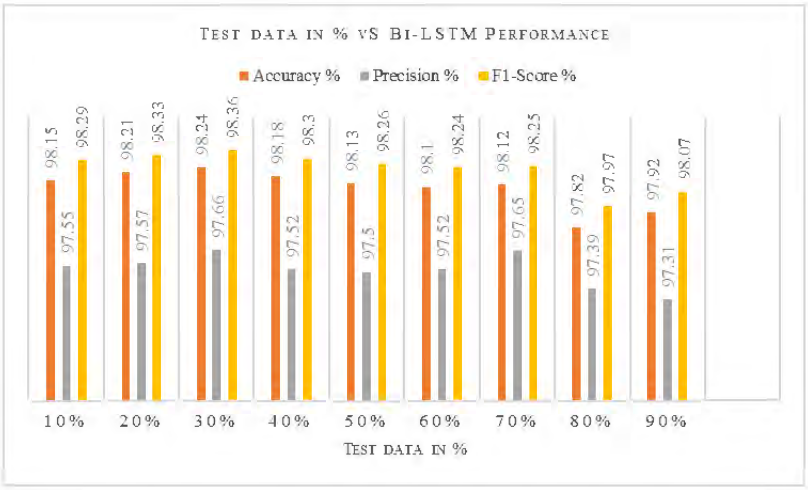

Fig. 5.5. Test data size in % vs Bi-LSTM model performance.

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 5.2. and the plot is shown in Figure 5.5, where a 30% test split percentage results in the model's highest accuracy of 98.48% and f1-score of 98.57%.

C. 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 5.2. and the plot is shown in Figure 5.5, where a 30% test split percentage results in the model's highest accuracy of 98.48% and f1-score of 98.57%.

TABLE 5. 3
BATCH SIZE VS.BI-LSTM 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 5.3. 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.

D. 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 5. 4
EPOCHS VS BI-LSTM 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, which is shown in Table 5.4.

However, it is important to note that a larger epoch value increases the training time for the 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%.

E. Experiment: Model layers parameters Vs. accuracy

In 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. The results revealed 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. 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 5.5. 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.

F. 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 5.6. The network attack traffic in these datasets is classified into 'Denial of Service,' 'Probe,' 'Remote to Local,' and 'User to Root' [77]. 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 5. 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 is 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. The 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 5. 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 a 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 5. 8
BI-LSTM WITH SMOTE TECHNIQUES 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				

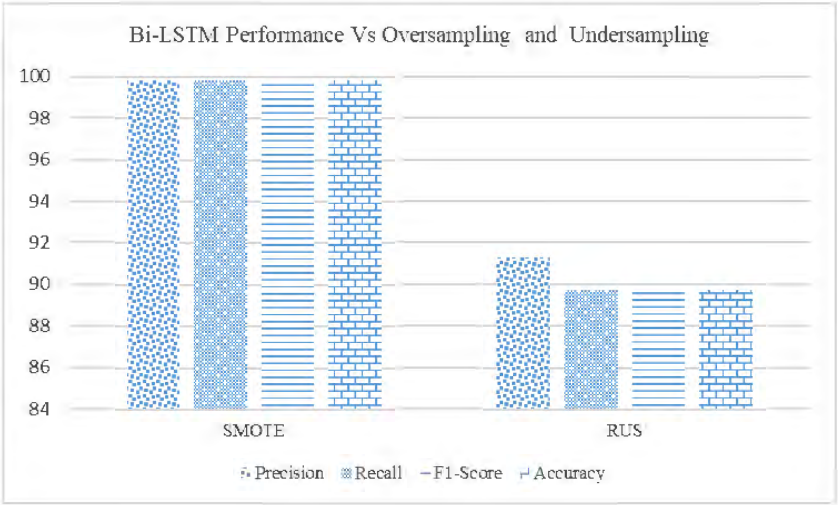

Fig. 5.6. Bi-LSTM performance vs over-sampling and under-sampling.

In the 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 5.7. The program execution time is short as compared with oversampling. Table 5.8. 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 5.6 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.

VI. CONCLUSION

The highest performance is achieved during network traffic anomaly detection using the bidirectional LSTM model. The combination of tuned different hyperparameters from the above experiment's 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 [61] to prove that the 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% [61]. 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 [61]. 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.

CHAPTER 6. ENHANCING THE NETWORK ANOMALY DETECTION USING CNN-BIDIRECTIONAL LSTM HYBRID MODEL AND SAMPLING STRATEGIES FOR IMBALANCED NETWORK TRAFFIC DATA

Abstract- The cybercriminal utilized the skills and freely available tools to breach the networks of internet-connected devices by exploiting confidentiality, integrity, and availability. Network anomaly detection is crucial for ensuring the security of information resources. Detecting abnormal network behavior poses challenges because of the extensive data, imbalanced attack class nature, and the abundance of features in the dataset. Conventional machine learning approaches need more efficiency in addressing these issues. Deep learning has demonstrated greater effectiveness in identifying network anomalies. Specifically, a recurrent neural network model is created to recognize the serial data patterns for prediction. We optimized the hybrid model, the convolutional neural network combined with Bidirectional Long-Short Term Memory (BLSTM), to examine optimizers (Adam, Nadam, Adamax, RMSprop, SGD, Adagrad, Ftrl), number of epochs, size of the batch, learning rate, and the Neural Network (NN) architecture. Examining these hyperparameters yielded the highest accuracy in anomaly detection, reaching 98.27% for the binary class NSL-KDD and 99.87% for the binary class UNSW-NB15. Furthermore, recognizing the inherent class imbalance in network-based anomaly detection datasets, we explore the sampling techniques to address this issue and improve the model's overall performance. The data imbalance problem for the multiclass network anomaly detection dataset is addressed by using the sampling technique during the data preprocessing, where the random over-sampling methods combined with the CNN-based BLSTM model

outperformed by producing the highest performance metrics, that is, detection accuracy for multiclass NSL-KDD and multiclass UNSW-NB15 of 99.83% and 99.99% respectively. Evaluation of performance, considering accuracy and F1-score, indicated that the proposed CNN BLSTM hybrid network-based anomaly detection outperformed other existing methods for network traffic anomaly detection. Hence, this research contributes valuable insights into selecting hyperparameters of deep learning techniques for anomaly detection in imbalanced network datasets, providing practical guidance on choosing appropriate hyperparameters and sampling strategies to enhance model robustness in real-world scenarios.

Keywords—Convolutional neural network, CNN-BLSTM, data imbalance, network anomaly detection, NSL-KDD, random over sampling, random under sampling, UNSW-NB15.

I. INTRODUCTION

As technology undergoes rapid advancements, the transmission of information has transformed significantly, adopting various methods such as wired, wireless, or guided networks. This evolution in network technology is pivotal to people's daily activities. Whether it is communicating with others, accessing online resources, or sharing information, the efficiency and security of these interactions depend heavily on the underlying network infrastructure. A system attains security when it effectively maintains the three essential notions of computer information security: Availability, Integrity, and Confidentiality (CIA). In essence, information security involves safeguarding information from unauthorized entities and protecting against illegal access, use, disclosure,

reformation, recording, or destruction of data. Confidentiality guarantees that the information is accessible only to individuals or systems.

In information network technology, encryption methods and access controls prevent unauthorized users from gaining access to sensitive data during transmission. Information integrity guarantees that data remains unaltered during transmission. In the context of information network technology, this involves implementing mechanisms to detect and prevent unauthorized modifications to data, ensuring that the information received is the same as what was sent.

Availability ensures that information and resources are available and accessible when needed. In an information resources and security environment, availability involves designing robust and reliable systems that can withstand potential disruptions, whether they are due to technical failures or malicious attacks. The overarching goal of information security is to safeguard information from unauthorized access and malicious activities. This includes preventing unauthorized individuals or systems from gaining access to sensitive data, ensuring that information remains unchanged and reliable during transmission, and guaranteeing that information and resources are available when needed.

Measures to achieve information security encompass a range of strategies, including encryption to protect data confidentiality, checksums or digital signatures to ensure data integrity, and redundancy and fault-tolerant systems to enhance availability. Additionally, access controls, firewalls, and intrusion detection are commonly utilized to fortify the security posture of networked systems, mitigating the risks associated with information resources.

A traditional network cannot be fully protected by relying solely on a firewall and antivirus software. These security measures identify predefined anomalous activities and establish the rule to prevent those unusual events by the cyber expert. In anomaly detection, outliers and anomalies are occasionally employed interchangeably. This approach finds extensive use across diverse domains, such as commercial, network attack detection, health systems monitoring, credit card fraud transaction detection, and identifying faults in mission-critical infrastructure systems. Anomaly detection is crucial in cybersecurity, providing robust protection against cyber adversaries. Ensuring safeguard network resources is essential to safeguard the organization from cyber threats.

Anomalies are categorized into point, contextual, and collective types based on the results generated by the detection method [48]. Point anomalies occur when a specific activity diverges from the typical rules or patterns. Contextual anomalies involve unusual patterns within a particular circumstance that consistently differ from numerous normal activities. Collective anomalies occur when a group of related instances exhibit anomalous behavior compared to the normal activity dataset.

Intrusion detection techniques can be broadly classed into two main types: Signature-based Intrusion Detection System (SIDS) and Anomaly-based Intrusion Detection System (AIDS). Anomaly detections, in contrast, are classified according to their origins, resulting in network-based and host-based intrusion/anomaly detection systems. Detecting anomalies in data is facilitated by employing labels to differentiate between normal and abnormal occurrences. There are three fundamental approaches to detecting anomalies: supervised, semi-unsupervised, and unsupervised methods. In the supervised approach, the system is trained on labeled data, distinguishing between normal and

anomalous instances. On the other hand, unsupervised methods detect anomalies without prior labeling, relying on deviations from established patterns. Semi-supervised techniques combine elements of both, using labeled and unlabeled data for training. AIDS overcomes the drawbacks of SIDS by utilizing ML, statistical-based, or knowledge-based methods to model normal behaviors. However, it is worth noting that anomaly-based detection may produce false results due to alterations in user habits.

Numerous traditional machine learning algorithms favor shallow learning methodologies, giving significant importance to feature engineering designed for smaller data. The feature engineering phase needs more processing time and domain expertise to create pertinent features and eliminate unrelated ones from anomaly detection algorithms. The effectiveness of anomaly detection is intricately tied to feature engineering and data preprocessing implementation. Traditional machine learning methods, characterized by simplicity, low resource consumption, and subpar performance in areas like vision, language processing, and image translations, underscore the limitations of these approaches.

CNN is predominantly employed for image signals, leveraging its architecture to effectively capture and analyze visual information. Individual neurons play a key role in reducing the dimensionality of the network's features in the lower layers of a CNN. These neurons are adept at identifying essential small-scale features within the images, including boundaries, corners, and variations in intensity. The CNN network links lower-level features to produce more complicated features in the upper layers, encompassing fundamental shapes, structures, and partial objects. The ultimate layer of the network amalgamates these lower features to generate the output results.

The functioning of a long short-term memory differs from that of a CNN due to its specific design to safeguard long-range info within a sequential order. Unlike CNNs, LSTMs are crafted to remember and store information over extended sequences, avoiding the loss of crucial details. In the case of BLSTM, an additional LSTM layer is incorporated, introducing a reversal in the information flow direction. This architectural enhancement addresses challenges related to vanishing gradients, ensuring more effective training by considering information from both forward and backward directions in the sequence.

Data imbalance, including network anomaly detection, is a common challenge in ML applications. In network traffic anomaly detection, data imbalance refers to the unequal distribution of normal and anomalous instances in the dataset for training the detection model. Anomalies in network traffic are typically rare incidents compared to normal activities, leading to imbalanced data.

The deep learning approach addresses issues found in conventional machine learning. The effectiveness of the deep learning-based anomaly detection algorithm relies on factors such as the NN architecture, #hidden layers, activation functions, batch size, and the number of epochs utilized during DL model testing, training, and validation. The careful selection of these factors, including hyperparameters and the architecture of NN in deep learning, is crucial for enhancing the detection accuracy of network traffic anomaly detection. The essential selection of ML or DL models overcomes the class imbalance problem. The ensemble method, which combines more different individual models, requires longer training time and consumes more resources. The sampling method generates random data or deletes the random data based on the implemented sampling methods to create the balanced form of the final dataset, which is efficient in dealing with the imbalanced dataset.

II. LITERATURE REVIEW

The rapid increase of information and technology has led to widespread connectivity of numerous end terminals to the internet and networks. Those smart terminals contribute to generating substantial volumes of data, commonly referred to as big data. This huge flood of data is a valuable resource for analysis and insights. Machine learning and deep learning algorithms come into play to extract meaningful information from this vast data pool. The daily growth of big data presents difficulties for conventional machine learning algorithms, necessitating thorough feature extraction and discovery. DL substantially increases anomaly detection and model performance. Nevertheless, the dataset's attributes and features, hyperparameters in deep neural networks, and the structure of neural networks are pivotal elements that impact the efficacy of identifying anomalies in network-based IDS.

Conventional machine learning strongly relies on intricate and time-consuming feature engineering, often impractical for real-time applications. In the [65] study, the authors proposed an approach for payload classification utilizing CNN and RNN to detect attacks, achieving detection accuracies of 99.36% and 99.98% on the DARPA98 network data, respectively. CNN methods discern specific grouping patterns through convolution around input neighborhoods, while RNN works on sequences by calculating correlations between previous and current states. In another [66] study, class imbalance was handled utilizing a CNN with a Gated Recurrent Unit (GRU) hybrid model. To address the data class imbalance and feature redundancy, they used a hybrid sampling technique that integrates Pearson Correlation Analysis (PCA), repeated edited nearest neighbors, Random Forest (RF), and adaptive synthetic sampling. With the detection accuracies of 99.69%, 86.25%,

99.69%, and 99.65% on the NSL-KDD, UNSW_NB15, and CIC-IDS2017 datasets, respectively, their CNN-GRU model performed better.

The research authors [49] proposed using an Adaptive Synthetic Sampling (ADASYN) technique in a DL-based network intrusion detection system to overcome dataset imbalance. On the NSL-KDD network data, they used an autoencoder to reduce dimensionality. The CNN-BLSTM hybrid DL method obtained the greatest F1 score (89.65%) and accuracy (90.73%). To address problems resulting from data in class imbalance and heterogeneous data distribution across various information sources, the research [67] used convolutional neural networks with federal transfer learning. The UNSW-NB15 multiclass network dataset produced an average detection accuracy was 86.85% for the model.

In [53], the researcher addressed data class imbalance on network datasets: NSL-KDD, KDD99, and UNSW-NB15 datasets using heterogeneous ensemble-assisted ML methods for binary and multi-class network intrusion detection. Using the NSL-KDD dataset, the model showed a 96.2% AUC and a true positive rate (TPR) of 94.5%. The authors of [54] discovered that ML classifier performance increased with the decrease in target classes. Conventional ML approaches, such as Naïve Bayes, Random Forest, J48, Bagging, Adaboost, and BayesianNet, were used to investigate this idea on three network traffic-based intrusion datasets: KDD99, UNSW-NB15, and CIC-IDS2017_Thursday.

In a study [68], the authors suggested a method for achieving network intrusion classification with low computing cost, creating a group of target classes based on the nature of network traffic. They created cluster characteristics for each group using K-means on the KDD99 network dataset, resulting in a detection accuracy of 98.84%. However, the

intrusion detection model accuracy for user2root (U2R) is notably low at 21.92%, impacting overall performance. In [69], authors employed a hybrid method, combining CNN and LSTM, to enhance model classification accuracy, achieving 96.7% and 98.1% on CIC-IDS2017 and NSL-KDD network data, respectively.

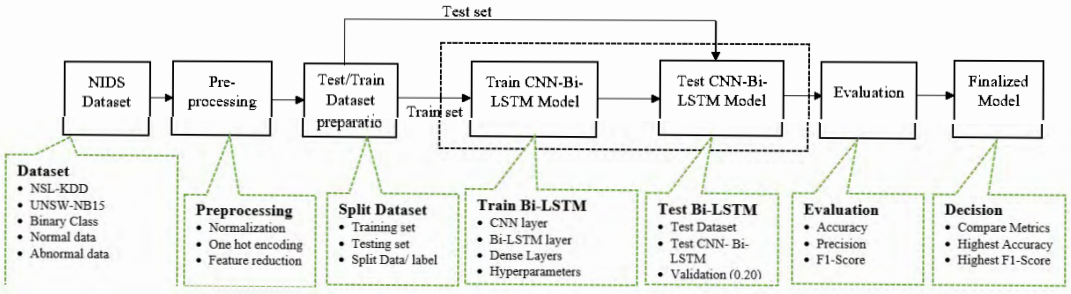

Fig. 6. 1. CNN Bidirectional LSTM model block diagram.

In the study [70], CNN and LSTM combined to create a hybrid model was proposed to enhance network intrusion detection model facilities for advanced metering infrastructure through cross-layer features combination. This method achieved the highest network intrusion detection accuracy of 99.79% on NSL-KDD and 99.95% on KDD Cup99 but with limited user2root (U2R) detection capabilities. Similarly, in [71], authors employed a hybrid method of combining CNN and LSTM to improve model network intrusion detection by capturing additional network traffic data's spatial and temporal features.

In [72], the researchers implemented a hybrid technique based on the mean control of the CNN and BLSTM to address issues of conventional data pre-processing and imbalanced numerical distribution of class instances in the NSL-KDD, achieving the optimal detection accuracy of 99.10%. However, the accuracy for the minority traffic data class remains suboptimal. Using a different methodology, the authors [73] created a DL model that combined CNN and BLSTM to learn temporal and spatial characteristics.

Accuracy levels on the binary class UNSW-NB15 were 93.84%, and binary NSL-KDD of 99.30%.

Data was preprocessed using one-hot encoding and min-max normalization by authors in [74], which achieved an accuracy of 96.3% on CNN and Bi-LSTM hybrid methods on the multiclass NSL-KDD dataset. Using preprocessed on given NSL-KDD data, researchers in [59] applied the hybrid model using CNN and BLSTM algorithm with a 95.4% accuracy rate. A bidirectional LSTM model was used by the authors in their study [78] for the binary NSL-KDD dataset with the highest accuracy of 98.52%. Using a Bidirectional LSTM deep learning model, authors [61] got 99% accuracy on UNSW-NB15 and KDDCUP-99, which is an exceptional achievement. But a lot of the models that are now in use need help effectively identifying uncommon (rare) attack types, especially user2root (U2R) and remote2local (R2L) attacks, which frequently have poorer detection accuracy as compared with other network attack types.

To overcome the difficulties found in the above literature review, authors in [62] presented a Bi-LSTM-based network intrusion detection system on the NSL-KDD dataset, which offered a binary classification accuracy of 94.26%. Furthermore, the authors proposed a Bi-directional GAN-based method [51] for the NSL-KDD and CIC-DDoS2019 datasets. The bidirectional GAN model demonstrated strong performance with an f1 score and detection accuracy of 92.68% and 91.12%, respectively, on the unbalanced NSL-KDD dataset.

In the research study [78], [76], the Authors used the hyperparameters tuning to obtain the best model performance on network intrusion detection datasets, including NSL-KDD and UNSW-NB15. In [79], the Authors implement the BLSTM model combined with

random over-sampling strategies, which produces a high anomaly detection accuracy of 99.83% for multiclass imbalance network anomaly datasets NSL-KDD dataset.

The deep learning model discussed in [65] and [66] overcomes challenges traditional machine learning encounters in anomaly detection. While the CNN standalone model is unsuitable for sequential data preprocessing, and RNN requires complex data preprocessing, this model effectively addresses these issues. Data imbalance problems are tackled in [49], [67], [53], and [54]. Feature engineering emerges as a critical factor in enhancing the accuracy of both ML and DL models. Much research has been conducted on feature engineering, with studies focusing on attribute grouping found in [68], [69], [70], [71], The BLSTM, which brings together two distinct LSTMs to allow input processing in both directions (from the past to the future and vice versa), is implemented in [72], [73], [74], [59], [61], [62], and [51] to improve the accuracy of network anomaly detection models.

Most of the researchers mentioned above concentrate on enhancing the detection accuracy of conventional or ML DL models and employ ensemble methods for feature engineering to address data imbalance. However, there needs to be more emphasis on exploring hyperparameter selection in DL-based models, determining the train-test split ratio, and defining the architecture of DNN. Some researchers need to elaborate on adopting these values in their studies. Subsequently, this research addressed these limitations in network traffic anomaly detection systems. We experimented using binary and multiclass versions of the UNSW-NB15 and NSL-KDD. Our focus includes investigating the performance comparison between random under-sampling and over-sampling to identify superior methods for imbalanced network data.

The contributions of our research effort in the area of network anomaly detection and imbalanced datasets are listed as:

a. Examining the impact of CNN and BLSTM neural network architecture and performance for binary/multi-class datasets, specifically NSL-KDD and UNSW-NB15.
b. Exploring the model performance of hyperparameters on binary and multi-class network datasets, namely UNSW-NB15 and NSL-KDD.
c. Exploring the enhancement of CNN Bi-LSTM by varying memory elements and numbers of layers of NN.
d. This study's interest is developing and implementing a CNN Bi-LSTM hybrid model for network anomaly detection, achieving high accuracy rates of 98.27% on NSL-KDD binary data and 99.87% on UNSW-NB15 binary data.
e. Exploring the network anomaly detection model based on CNN Bi-LSTM using UNSW-NB15.
f. Investigating the random sampling methods for imbalanced data with detection accuracy greater than 99.83% for NSL-KDD multiclass data and 99.99% for the UNSW-NB15 multiclass dataset.

The rest of this chapter unfolds: Section 3 delineates the system model and individual blocks comprising our CNN Bi-LSTM hybrid approach. Section 4 elucidates the experimental setup, experimental results, and discussion of the findings, and section 5 encapsulates the conclusion of this research.

III. NETWORK ANOMALY DETECTION MODEL DESCRIPTION

The complete proposed model comprises the following steps:

1. Network traffic-based data collection
2. Data pre-processing and cleaning
3. Training and testing data preparation
4. CNN BLSTM model preparation
5. Train and test model
6. Evaluation of CNN BLSTM model
7. Compare the model and decision-making

The CNN BLSTM-based model's entire implementation schematic is displayed in Fig. 6.1. The ensuing sections offer a thorough explanation of the approaches mentioned previously. The components of CNN and BLSTM layers and the intricate architecture of neural networks are seen in Fig. 6.2.

A. Network traffic-based data collection.

Numerous datasets are accessible for research in network intrusion detection systems. Examples include the KDD Cup99, Kyoto 2006+, NSL-KDD, CICIDS2017, UNSW-NB15, and several others, providing valuable resources for intrusion detection research. During this research, the UNSW-NB15 and NSL-KDD datasets are specifically employed.

NSL-KDD KDDTrain+ [64] originates from the DARPA KDD99 dataset, with the elimination of noise and undesired data. This dataset encompasses the complete NSL-KDD training set, including labels denoting attack types and difficulty levels. Comprising 41 features, it delineates five different attack classes: “Normal,” “DoS,” “Probe,” “R2L,” and “U2R.”

NSL-KDD represents a refined form of the KDD99 data, free from duplicate records in the training set and the test sets. Each entry in the dataset consists of 42 attributes, with 41 of them related to the input traffic and the final label indicating whether the traffic is normal or abnormal (target). The KDDTrain+ dataset encompasses 125,973 data entries, while the KDDTest+ dataset consists of 22,544 data entries utilized in this research work. Table 6.1 documents the detailed information regarding the traffic and data information [77].

TABLE 6. 1 DETAILS OF NSL-KDD DATA

Traffic	KDDTrain+	KDDTest+
R2L	995	2,885
U2R	52	67
DoS	45,927	7,460
Normal	67,343	9,711
Probe	11,656	2,421
Total	125,973	22,544

Similarly, The Australian Centre for Cyber Security (ACCS) cybersecurity research team constructed the UNSW-NB15 dataset [75], unlike KDD99 and NSL-KDD, which is a recently developed network intrusion dataset created by IXIA PerfectStrom tools within the Cyber Range Lab of the ACCS, this dataset consists of approximately 100GB of PCAP files capturing raw network traffic flows between two hosts either server to client or vice versa. The Argus and Bro_IDS tools and 12 other algorithms generated 49 features accompanied by class labels. Numerous records were utilized to construct the training and testing sets, where UNSW_NB15_training-set and UNSW_NB15_testing-set were used during this research work. The training set comprises 175,341 records, while the testing set comprises 82,332 records, encompassing various attacks and normal network activity. Table 6.2 shows detailed information regarding the attacks and normal traffic.

TABLE 6. 2
DETAILS OF UNSW-NB15 DATA

Network Traffic	testing-set.csv	training-set.csv
Exploits	11,132	33,393
Generic	18,871	40,000
Worms	44	130
Fuzzers	6,062	18,184
DoS	4,089	12,264
Reconnaissance	3,496	10,491
Analysis	677	2,000
Backdoor	583	1,746
Shellcode	378	1,133
Normal	37,000	56,000
Total	82,332	175,341

The KDDTrain+ and KDDTest+ subsets of the NSL-KDD dataset were employed in our research experiment—likewise, experiments involved using training-set.csv and testing-set.csv from the UNSW-NB15 dataset.

B. Data pre-processing and cleaning

NSL_KDD data is an improved version of the KDD99 dataset; minimum work is required for data preprocessing. The downloaded separate data files are used to test and train the model. The target class is initially isolated from the training and testing datasets to create the class labels. From the remaining attributes, numerical features and three categorical features—"protocol_type", "service", and "flag" are extracted. The categorical features undergo conversion into numerical values using dummy one-hot encoding techniques, while the numerical attributes are standardized using standard Scalar methods. Afterwards, both types of feature sets are combined into a unified data frame, yielding the final data sets for training and testing. One hot encoding generates one binary variable for each individual categorial value. The dummy encoding is similar to one hot encoding and converts the categorical values into numeric binary values. The dummy encoding represents N categories using N-1 binary variable. Let's say we have three categories of traffic "protocol_type,"

“service,” and “flag” that are going to be dummy encoded as [1 0], [0 1], and [0 0], respectively. The standard scalar converts the numeric values so that the data standard deviations become 1.

Since there are different types of services present in the KDDTrain+ dataset and KDDTest+ dataset, the one hot encoding produces unequal numbers of features. The KDDTrain+ dataset contains 126 features, while the KDDTest+ includes a total of 120 features after the implementation of one hot encoding. Those additional features “service_aol,” “service_harvest,” “service_http_2784”, “service_http_8001”, “service_red_i,” and “service_urh_i” are inserted into the KDDTest+ dataset after finding the exact location where those features reside into the KDDTrain+ dataset. We preserved the attacks_types and difficulty_level features because those features are highly relevant to the target class and increase the model's efficiency.

The UNSW-NB15 dataset was divided into two sets for training and testing purposes: UNSW_NB15_training-set and UNSW_NB15_testing-set. The UNSW_NB15_training-set comprises 175,341 entries, while the UNSW-NB15_testing-set contains 82,332 entries, encompassing various attacks and normal data. Initially, the features on this dataset are 49. First, those categorical attributes are changed into numeric using dummy one hot encoding. All numerical attributes are applied to the standard scalar normalization method. After preprocessing the numeric and categorical features, 192 features for UNSW_NB15_testing-set data and 196 features for UNSW_NB15_training-set data were generated. Again, here we are taking two sets of data: one we can use for training and the other for testing or vice versa. The categorical values of data entries are not the same

for both datasets; hence, the one hot encoding produces unequal numbers of features on both data sets after preprocessing.

Some features generated from one hot encoding, such as state_ACC and state_CLO, are not included in the UNSW-NB15_training-set. Similarly, proto_icmp, proto_rtp, state_ECO, state_PAR, state_URN, and state_no features are not included on UNSW_NB15_testing-set. The empty features columns are added in the exact column location of those missing features on the respective dataset, generating 198 features plus one target class.

C. Training and testing data preparation

In experiments concerning the binary NSL-KDD dataset, the training and testing datasets were created using a split ratio. The train-test split approach assesses the performance of machine learning algorithms in making predictions from data that wasn't part of the training set. We opted for a 70:30 split ratio to generate the train and test dataset. For the CNN BLSTM hybrid model, 70% of KDDTrain+ was used to train, and the remaining data was used to test the model for binary NSL-KDD data.

A similar split percentage was employed in the binary class UNSW-NB15, using the "UNSW_NB15_training-set." In the case of multiclass experiments for UNSW-NB15 and NSL-KDD, two distinct files were selected—one subset for training the CNN-BLSTM model and another for testing. Detailed information regarding this split is provided in the respective experimental sections.

D. CNN BLSTM model

CNN is a forward DNN designed for image signal and classification. CNN comprises three primary layers: the convolutional, the pooling, and the fully connected

layers. The convolutional layer is the main component of CNN and uses the convolutional operation to grab the various features from the image signal. Then, the number of pooling layers extracts features, and a fully connected layer employs the output from the preceding layer for classification. Combining convolutional layers with pooling layers is responsible for feature extraction, while the final fully connected dense layer is utilized for classification purposes. CNN also involves various hyperparameters, including the number of filters, stride, zero-padding, pooling layers, and others.

An RNN is an artificial NN designed to manage sequential data by integrating feedback loops into its structure. Diverging from conventional feedforward neural networks that linearly handle input data, RNNs feature connections forming loops, enabling them to retain a memory of past inputs and utilize that information to impact the current output. The memory in an RNN serves as a short-term storage, allowing the network to retain information about past events and use it to make predictions about future events. This is especially valuable in applications where context and temporal relationships are essential. Machine learning issues, including speech recognition, language processing, and picture categorization, have been resolved with RNN.

Yet, traditional RNNs encounter challenges, notably needing help with learning long-term dependencies attributed to the vanishing or exploding gradient problem. Advanced RNN versions such as gated recurrent units (GRUs) and long short-term memory (LSTM) networks have been devised in response to these constraints. These architectures include mechanisms for selectively storing and retrieving information across extended sequences, enhancing their effectiveness in tasks that demand capturing long-term dependencies.

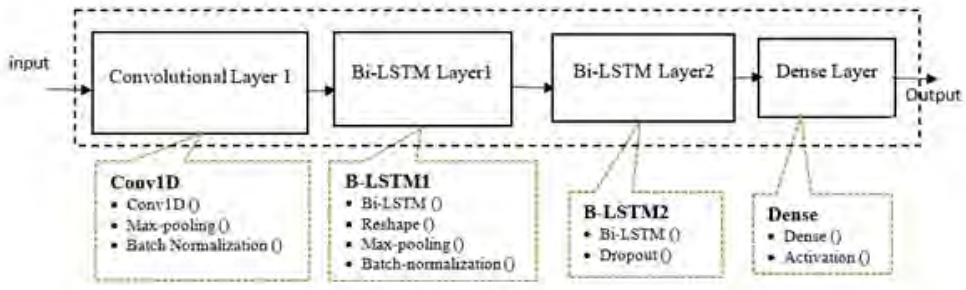

Fig. 6. 2. CNN BLSTM layer architecture.

LSTM handles the vanishing gradient in RNN. There is a memory block and three multiplicative units in LSTM. The input corresponds to the write operation, output to read and forget gates corresponds to the reset operations for cells that make up the LSTM architecture. By allowing LSTM memory cells to keep and access data for longer periods. Those multiplicative gates mitigate the vanishing gradient.

To process input in both directions—from the future to the past and from the past to the future—bidirectional RNN combines two independent RNNs. Both forward and backward LSTM networks make up the Bi-LSTM. The features extracted by the forward LSTM hidden layer point forward, whereas those extracted by the reverse LSTM hidden layer point backward. By taking finite sequences into account about earlier and later items, the bidirectional LSTM can anticipate or tag the sequence of each element. Two LSTMs processed in series—one from left to right and the other from right to left—produce this. The CNN and BLSTM hybrid models have several layers, each with a set of hyperparameters. Fig. 6.2. shows the CNN BLSTM's architectural layout.

E. CNN BLSTM model training

The CNN BLSTM model's neural network architecture is prepared for training. The datasets consist of two sets: one for training and the other for testing, or vice versa. The split

percentage determines how much data is allocated for training and testing when a single data set is present. The selection of hyperparameters for model training is conducted through various experiments involving fine-tuning epochs and batch size to enhance detection efficiency. Within the training data, 20% is designated for validating the CNN Bi-LSTM model.

F. Test the CNN BLSTM hybrid model and evaluation.

Deep learning (DL) and machine learning (ML) models offer performance consistency. After the CNN BLSTM model is built, the model is trained using the training dataset with specified hyperparameter values. These chosen hyperparameter values influence the training duration. Following training, the model can assess the unseen dataset to evaluate its performance. Hyperparameter selection lacks a predefined rule, allowing for random selection and subsequent fine-tuning through various experiments.

After the model testing, performance metrics are determined based on the type of ML model employed. In the case of the supervised machine learning model, ground truth values are utilized to measure the performance metrics on the test dataset. Various metrics, such as detection accuracy, precision, F1-Score, recall, program execution time, and Area under the ROC, are available to compare the model efficiency. Confusion metrics from Karas generate True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. In the context of a classification report, the terms "weighted" and "macro" refer to different strategies for computing metrics such as precision, recall, and F1-score across multiple classes. Macro-averaging computes the metric for every class separately before averaging them. This means that each class is treated equally in the computation, regardless of size. Macro-averaging gives the same weight to each class,

which can be useful when all classes are considered equally. Weighted averaging, on the other hand, takes the average of the metrics, but it weights each class's contribution based on its proportion in the dataset. In other words, classes with more samples have a greater impact on the average. W weighted averaging is especially helpful when working with unbalanced datasets—where certain classes may have substantially more instances than others.

The classification report provides a thorough summary of the model's performance metrics for the specified training and testing data sets. Lastly, to assess the performance of our CNN BLSTM hybrid model, the performance metrics are compared with the findings of earlier research publications.

G. COMPARE MODELS AND DECISION-MAKING.

Several sets of experiments were conducted to find the hyperparameter settings that yielded the best results. Following model testing and evaluation, choosing the best model pipeline from various options is part of the cognitive process of comparison and decision-making. Throughout this study, several sets of experiments are carried out to determine values for various CNN Bi-LSTM model hyperparameters to enhance the model's performance. To create an effective Bi-LSTM pipeline, it is necessary to decide on the hyperparameters, which include optimizers, number of epochs, batch, NN design, class size, and techniques of raw data preprocessing. This is achieved by evaluating performance metrics across multiple sets of experiments. The performance metrics of the Bi-LSTM model are then juxtaposed with previously published results for the binary/numerous class UNSW-NB15 and NSL-KDD. The class imbalance problem in the multiclass version of both NSL-KDD and UNSW-NB15 datasets was exposed with sampling data during the

preprocessing stages. The sampling methods randomly deleted on down-sampling and randomly generated data samples in over-sampling. This resulted in the balanced form of datasets to compare the CNN Bi-LSTM model performance.

IV. RESULTS AND DISCUSSION

To detect anomalies, intrusion detection uses a mix of DL and ML methods. The implementation of a network anomaly detection model is implied using Python script. Python has specialized packages for building machine learning models, including NumPy, Pandas, Keras, and Scikit-learn. Additionally, commonly used tools like Java, C#, WEKA, Visual C++, and MATLAB play vital roles in network anomaly detection systems. On the Jupyter Notebook platform, seed values are fixed to guarantee consistency in outcomes over several runs. Plots and tables representing the results of experiments are analyzed using the Microsoft Office suite. Every experiment is run on a Windows machine with an i7 processor and 16GB of RAM.

Python and the packages it is linked with keep version information used in all experiments. For example, TensorFlow 2.9.1, Keras 2.6.0, and Python 3.7.12 are used. Hyperparameters will be determined, performance will be evaluated across class sizes, and the efficacy of various sampling approaches will be assessed about the CNN BLSTM model for the multi-class and binary-class UNSW-NB15 and NSL-KDD. Detailed explanations of these experiments are provided in subsequent sections.

The architecture shown in Fig. 6.2 consists of a single 16-unit convolution layer that uses batch normalization and max-pooling. BLSTM neural network layer 1 contains 50 memory units; batch normalization, max-pooling, and reshaping come next. Bi-LSTM neural network layer 2 with 100 memory units and dropout is also available. The dense layer

consists of a sigmoid activation, and the final output is obtained. The detection accuracy of the model is evaluated through a series of tests involving the adjustment of optimizers, learning rate (LR), number of epochs, batch size, and dropout rate. As explained below, the UNSW-NB15 and NSL-KDD binary/multiclass network traffic datasets are used for these investigations.

A. Experiment: Model performance Vs. Optimizers

In the context of ML and DL, an optimizer is an algorithm or method used to adjust the parameters of a model to minimize or maximize a certain objective function. The performance of an optimizer is crucial in training machine learning models because it determines how well the model learns from the data. Choosing the optimizer is essential during the training of the CNN BLSTM model, as it significantly contributes to expediting results for the machine learning/deep learning model.

TensorFlow offers nine optimizers (Ftrl, Nadam, Adam, Adadelta, Adagrad, gradient descent, Adamax, RMSprop, and Stochastic Gradient Descent (SGD)) based on the optimizer's methods. The choice of optimizer can significantly impact the training performance of an ML model. Optimizers may converge at different rates or achieve different final accuracies on a given task. An optimizer's performance may be influenced by the model's architecture, the dataset, and the hyperparameters employed.

It is common practice to experiment with several optimizers to determine which combination of optimizers and hyperparameters is optimum for a given task. Additionally, some optimizers may perform better on certain types of neural network architectures or for specific data types. In summary, the relationship between the optimizer and machine learning performance is crucial, and choosing the right optimizer is an important part of the

model training process. It often involves experimentation and tuning to find the optimal combination for a given task.

The model used in the experiment comparing Optimizers versus Accuracy has a 20% dropout rate and the Relu activation function. To determine the best optimizer for our CNN-BLSTM model, seven optimizers, including Nadam, Ftrl, SGD, Adam, RMSprop, Adagrad, and Adamax, were tested. Based on the model performance metrics for UNSW-NB15 and NSL-KDD binary data, which are shown in Table 6.3, it was found that the Nadam optimizer performed best for NSL-KDD. In contrast, the Adam optimizer produced the best accuracy for the UNSW-NB15 dataset. Interestingly, although both optimizers used the same model architecture, they performed differently for both Network Intrusion Detection System (NIDS) datasets.

TABLE 6. 3
CNN BI-LSTM PERFORMANCE VS OPTIMIZER FOR BINARY CLASS NSL-KDD

Number of epochs = 10, Batch = 256, NSL-KDD_C2 and UNSW-NB15_C2				
Optimizer	**ACC-NSL**	**F1-NSL**	**ACC-UN**	**F1-UN**
Ftrl	53.47	69.68	80.99	80.99
RMSprop	97.87	98.01	97.93	98.46
Adamax	97.65	97.78	95.33	96.51
Adam	98.02	98.16	**99.15**	**99.38**
Adagrad	96.98	97.21	94.04	95.62
SGD	97.74	97.91	99.14	99.37
Nadam	**98.13**	**98.26**	99.11	99.34
ACC: Accuracy in %, F1: F1Score in %, NSL: KDD-NSL, UN: UNSW_NB				

TABLE 6. 4
CNN BI-LSTM PERFORMANCE VS OPTIMIZER ON MULTICLASS NSL-KDD

Model Performance Vs. Optimizer on NSL-KDD Multiclass Datasets			
Epochs=10, Batch_size= 512, Training_data = KDDTrain+, Testing_data = KDDTest+, Multiclass=5			
Optimizers	**Accuracy %**	**wt_Precision %**	**wt_F1score %**
Adam	**88.46**	**88.87**	**88.23**
RMSprop	85.49	87.15	82.84
Nadam	84.79	86.97	82.45
SGD	82.86	84.99	77.12

Adamax	82.6	86.99	82.72
Adagrad	75.65	67.01	69.94
Ftrl	43.08	18.56	25.94

TABLE 6. 5
CNN BI-LSTM PERFORMANCE VS OPTIMIZER ON MULTICLASS UNSW-NB15

CNN Bi-LSTM Performance Vs. Optimizer on UNSW-NB15 Multiclass Datasets			
Epochs=15, Batch_size= 512, Training data=UNSW-NB15Train82332, Testing_data = UNSW-NB15Test175341, Multiclass=10			
Optimizers	**Accuracy %**	**wt_Precision %**	**wt_F1score %**
SGD	**89.84**	**87.49**	**88.01**
Adam	87.21	87.47	85.96
Nadam	84.59	84.48	83.38
RMSprop	79.3	75.71	76.85
Adamax	76.84	76.37	74.74
Adagrad	70.82	63.28	62.05
Ftrl	31.94	10.20	15.46

The selection of the optimizers depends on the combination of the different hyperparameters and NN architecture of the CNN BLSTM model. Popular optimization algorithm Adam combines concepts from RMSprop and momentum. It adapts the learning rates of individual parameters and is widely used in deep learning. An Adam extension that uses the Nesterov Accelerated Gradient (NAG). NAG involves looking ahead in the direction of the momentum before computing the gradient that combines the benefits of Adam and Nesterov momentum. Fig 6.3. shows the accuracy comparison for NSL-KDD and UNSW15.

Tables 6.4 and 6.5 show the comparative performance metrics of the multi-class NSL-KDD and UNSW-NB15. The same optimizer does not provide the same performance for a similar dataset. The hyperparameters and datasets used to test and train the CNN-based BLSTM model are provided in Tables 6.4 and 6.5. For the NSL-KDD multiclass dataset,

Adam performed better than SGD, whereas, for the UNSW-NB15 multiclass dataset, SGD performed better than other optimizers.

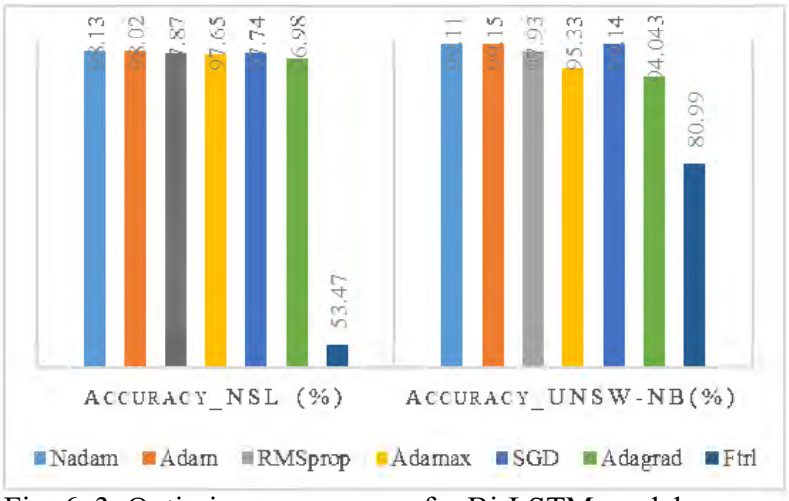

Fig. 6. 3. Optimizer vs accuracy for Bi-LSTM model.

B. Experiment: Learning rate Vs. model performance

The learning rate, a positive scalar multiplied by gradient descent gradient, controls the step size in parameter space. A higher rate facilitates faster convergence but raises the risk of overshooting and oscillation. On the other hand, a lower rate ensures stability but may demand more iterations for convergence.

With optimizers chosen from the preceding Experiment 6.A., the same CNN BLSTM model neural network architecture is used to determine the ideal learning rate to enhance the model performance. The NSL-KDD binary data is preprocessed from the subset of the KDDTrain+ dataset, and the split ratio splits the data for training and testing. The learning rate determines the rate at which new weights are added to neural network models. The other hyperparameters remain constant throughout this experiment while the learning rates are adjusted to optimize the model's accuracy. Table 6.6 displays a comparison of learning rate with CNN BLSTM model performance. The model performs best on the

UNSW-NB15 binary data and the NSL-KDD binary dataset, achieving a learning rate of 0.01 and 0.0002, respectively. The same learning rate provides different model performances.

TABLE 6. 6
CNN BI-LSTM MODEL LEARNING RATE VS PERFORMANCE

Epochs size = 10, Batch = 256, KDD_C2 (Nadam), UNSW-NB15_C2 (adam)				
LR	**ACC-NSL**	**F1-NSL**	**ACC-UN**	**F1-UN**
0.01	97.49	97.67	**99.67**	**99.76**
0.001	98.16	98.29	99.54	99.66
0.0001	98.06	98.20	95.81	96.85
0.0002	**98.18**	**98.3**	97.9	98.44
0.0003	98.14	98.27	98.44	98.86
0.0004	97.97	98.11	99.13	99.35
0.0005	98.11	98.25	99.09	99.32
LR: Learning rate, ACC: Accuracy in %, F1: F1Score in %, UN: UNSW_NB				

C. Experiment: Model dropout rate Vs. model performance

The phrase "dropout rate" in machine learning usually refers to a regularization method that neural networks employ to avoid overfitting. When a model becomes overfit, it can have poor generalization on new, unknown data because it has learned the training set too well, including its noise and outliers. During training, randomly selected neurons (units) in the neural network are "dropped out" or omitted temporarily. This means these neurons do not contribute to the forward or backward pass during a particular iteration of training. The probability of a neuron being dropped out is called the dropout rate. The dropout rate is one hyperparameter that must be determined before training the model.

The CNN BLSTM model was tested and trained for both datasets using a batch size of 256 and 10 epochs. Different dropout rate values were used to evaluate the efficiency of the model. The model performed better than the others, with a 30% dropout rate on the UNSW-NB15 dataset; however, a 60% dropout rate worked better for the NSL-KDD. The

hyperparameter values, dropout rates, and corresponding performance metrics are presented in Table 6.7. The experimental results highlight the varying dropout rates for distinct datasets despite the similarity between the two datasets.

TABLE 6. 7
DROPOUT RATE VS CNN-BLSTM PERFORMANCE

Epochs size = 10, batch = 256, KDD_C2 (madam), UNSW-NB15_C2 (adam)				
DropOut %	**ACC-NSL**	**F1-NSL**	**ACC-UN**	**F1-UN**
0.1	98.10	98.24	97.44	98.15
0.2	98.02	98.16	98.98	99.25
0.3	98.16	98.29	**99.87**	**99.9**
0.4	98.04	98.17	99.27	99.47
0.5	97.93	98.09	99.47	99.61
0.6	**98.21**	**98.33**	99.81	99.86
0.7	98.01	98.15	99.58	99.69
0.8	98.04	98.18	98.57	98.94
ACC: Accuracy in %, F1: F1Score in %, NSL: KDD-NSL, UN: UNSW_NB KDDTrain+, UNSW-NB15 training.csv binary with test-train split				

The batch size is a hyperparameter in machine learning that determines how many samples are used in a training iteration. The batch size represents the number of samples used in a single training iteration. Using a smaller batch size incorporates a limited number of data samples and results in a longer training time for the CNN Bi-LSTM model compared to a larger batch size. Throughout experimentation (Experiment A-C), the batch size is altered while maintaining other hyperparameters, such as a fixed number of epochs is 5, the learning rate of the optimizer, and the dropout rate values assigned to the model based on previous findings with the respective datasets.

TABLE 6. 8
CNN-BLSTM MODEL PERFORMANCE VS BATCH SIZE

Number of Epochs = 5, KDD_C2 (Nadam), UNSW-NB15_C2(adam)				
Batch	**ACC-NSL %**	**F1-NSL %**	**ACC-UN %**	**F1-UN %**
32	97.89	98.04	**99.40**	**99.55**
64	97.95	98.10	99.35	99.52
128	**98.06**	**98.20**	99.33	99.50
256	97.64	97.79	96.36	97.26
512	97.92	98.08	96.90	97.70

The dataset size, the amount of computing power available, and the specifics of the optimization issue can all influence the batch size decision. Experimenting with various batch sizes is a frequent way to determine which is most effective for a certain task. The experimental result in Table 6.8 demonstrates how the neural network's hyperparameter combinations affect performance. In this experiment, batch sizes of 128 for the binary NSL-KDD datasets and 32 for the binary UNSW-NB15 datasets for epochs 5 demonstrated the best performance of the CNN BLSTM model.

D. Experiment: Epochs Vs. model performance

An "epoch" in machine learning is one whole iteration through the training dataset a model goes through while training. The learning method processes the complete dataset throughout each epoch, modifying the neural network weights and parameters to reduce the error or loss function. A hyperparameter called epoch count determines an algorithm's running frequency over the full training dataset. The integer between one to infinity can be used as the epoch. Selecting smaller epoch values results in a longer training time for the model and vice versa. Underfitting, the ML model cannot identify the original patterns in the data, which can be caused by using too few epochs. However, an excessive number of epochs might cause overfitting, in which case the model becomes inattentive to new data and underperforms on previously unknown data.

The CNN BLSTM hybrid model performance for binary KDD-NSL and binary UNSW-NB15 with the different values of epochs are documented in Table 6.9. The performance increases with large values of epochs but is different for a while. After 75 epochs, the model performance decreases. The amount of data utilized for training and

testing, the size of the output class, and other hyperparameter combinations affect the epochs and performance of the machine learning/deep learning models.

TABLE 6. 9
EPOCHS VS CNN-BLSTM PERFORMANCE

Batch size = 256, NSL- KDD_C2 (Nadam)		
Number of Epochs	**Accuracy-NSL %**	**F1Score-NSL %**
2	95.48	95.94
10	98.13	98.26
25	98.21	98.33
50	98.20	98.33
75	**98.27**	**98.39**
100	98.26	98.39

The selection of epoch size to produce a superior performance on an imbalanced dataset is challenging. The binary dataset is more balanced than the multiclass network-based intrusion dataset. The experimental results in Table 9 are not the determining experiment for the number of epochs on multiclass NSL-KDD and UNSW-NB15 datasets. Hence, we experimented with and documented multi-class experimental results to determine the values of epochs where we can produce higher accuracy on the provided dataset. Tables 6.10 and 6.11 show the experimental results for multiclass datasets to investigate the values of epochs to make superior detection accuracy. In summary, while epoch size and class size are conceptually different, they can influence each other indirectly, especially when dealing with imbalanced datasets. Selecting the right number of epochs for a given problem is crucial, as is keeping an eye on how class sizes affect model performance.

TABLE 6. 10
CNN-BLSTM PERFORMANCE VS EPOCHS ON UNSW-NB15 MULTICLASS DATA

Batch=512, Optimizer=SGD, Training=UNSW-NB15Train.csv82332 testing data= UNSW-NB15Test.csv175341, Multiclass=10				
Epochs	**ACC**	**wt_Prec**	**wt_F1Score**	**Prg_exe_time**
10	**93.10**	**91.10**	**91.94**	**0.64**
25	83.09	79.94	79.69	1.17
50	86.4	82.47	83.35	2.24

75	87.1	86.46	85.24	3.12
100	90.04	88.36	88.01	4.24
150	82.23	81.49	80.82	6.36
200	81.13	78.65	78.84	7.95
ACC: Accuracy in %, wt_Prec: weighted Precision in %, wt_F1Score: weighted F1Score in %, Prg_exe_time: Program script run time in hr.				

Program execution time is the sum of the model's training and testing phases. The program execution time depends on various factors, such as the neural network's architecture, training and testing data size, class size, and combination of hyperparameters. We documented the performance of the CNN BLSTM hybrid model along with the program execution time. The higher the epochs result, the longer the program execution time. Table 6.10 and Table 6.11 show the multi-class NSL-KDD run for almost 8 Hrs. to complete training testing and evaluate the model for 200 epochs. A similar scenario for multiclass UNSW-NB15 dataset. Hence, selecting epoch and batch size is the trade-off with the model training, testing, and evaluation time. We found in Table 6.10 and Table 6.11 the different epoch sizes for NSL-KDD (outperform at epoch size 10) and UNSW-NB15 (outperform at epoch 100) during multi-class model performance.

TABLE 6. 11
CNN-BLSTM PERFORMANCE VS EPOCHS ON MULTICLASS NSL-KDD DATA

Batch=512, Optimizer= Adam, Data=KDDTrain+_125973, KDDTest+_22544, Multiclass = 5				
Epochs	**ACC**	**wt_Prec**	**wt_F1Score**	**Prg_exe_time**
10	86.21	88.13	83.85	0.45
25	86.64	88.01	84.1	1.08
50	86.64	88.7	84.78	2.15
75	87.11	88.39	84.84	3.31
100	**87.63**	**89.85**	**86.44**	**4.53**
150	87.22	89.66	85.81	7.03
200	86.84	90.61	86.41	8.76
ACC: Accuracy in %, wt_Prec: weighted Precision in %, wt_F1Score: weighted F1Score in %, Prg_exe_time: Program script run time in hr.				

E. Experiment: Imbalance data sampling Vs. performance

This experiment explores sampling techniques for imbalanced data to achieve a high detection rate. The researcher employed a combination of machine learning and deep learning algorithms, ensemble methods, and data sampling to tackle the challenge of data imbalance. This approach addresses the issue of fewer attacks compared to typical traffic data in the network intrusion detection dataset.

Sampling methods generate or delete random data from the dataset based on class data distribution. Random under-sampling and random over-sampling are two techniques used in imbalanced classification problems, where one class, usually the minority traffic class, is significantly under-represented compared to the other class(es). These methods are utilized to tackle class imbalance and enhance the efficacy of machine learning models. Random Under Sampling (RUS) involves randomly removing instances from the majority class until the distribution between the majority and minority classes is more evenly distributed. However, random over-sampling produces an equal distribution by randomly duplicating minority class instances or creating synthetic instances to increase the number of minority class instances.

Tables 6.12 and 6.13 provide the hyperparameter information and performance of this experiment's CNN BLSTM hybrid model. Table 6.12 compares the NSL-KDD multiclass dataset's performance when random over- and under-sampling is applied. After preprocessing multi-class NSL-KDD data, the training and testing datasets merge into a single file. Sampling is implemented on merged data, and a 70:30 split ratio is used to split data into train and test datasets.

TABLE 6. 12
CNN-BLSTM PERFORMANCE VS SAMPLING ON NSL-KDD

CNN BILSTM, Epochs= 25, Batch_size= 512, Data = combine (KDDTrain+KDDTest+) (sampling)				
Class	**Recall_RUS**	**F1_RUS**	**Recall_ROS**	**F1_ROS**
DoS	0	0	99.86	99.92
Probe	100	36.69	99.89	99.91
U2R	10	18.18	100	99.73
R2L	0	0	99.45	99.63
Normal	0	0	99.93	99.93
Wt_Average →	22.15	10.07	99.83	99.83
Macro_Avg→	22	10.97	99.83	99.83
Accuracy %	22.15		**99.83**	
[F1:F1Score, RUS: Random Under Sampling, ROS: Random Over Sampling] %				

Similarly, preprocessed training data and testing data files are merged into a single file to implement the sampling method on the UNSW-NB15 multiclass dataset. The sampled data is then split into training and testing datasets using a 70:30 train-test split ratio. During Random under-sampling, data instances are randomly deleted from the majority class, resulting in significant information loss. Deleting samples from the majority class results in a smaller sample, unsuitable for the deep learning model and worsens the model performance, which is found in the experiment and documented in Tables 6.12 and Table 6.13. Random over-sampling (ROS) helps prevent information loss, as none of the minority class instances are removed. It can be more effective when the amount of data in the minority class is limited.

TABLE 6. 13
CNN-BLSTM MODEL PERFORMANCE VS SAMPLING ON UNSW-NB15

CNN_BLSTM Epochs=25, Batch_size=512, data = combine (UNSW-NB15training-set_175341+UNSW-NB15testing-set_82332) sampling				
Class	Recall_RUS	F1_RUS	Recall_ROS	F1_ROS
Analysis	0	0	100	100
Backdoor	0	0	100	100
DoS	0	0	100	99.95
Exploits	100	17.51	99.91	99.95
Fuzzers	0	0	99.98	99.99

Generic	0	0	99.98	99.98
Normal	0	0	100	100
Reconnaissance	0	0	100	100
Shellcode	0	0	100	100
Worms	0	0	100	100
Weighted_Average	9.2	1.61	99.99	99.99
Macro Average	10	1.75	99.99	99.99
Accuracy (%)	9.20		**99.99**	
[F1:F1Score, RUS: Random Under Sampling, ROS: Random Over Sampling] %				

This method generates random data based on the data distribution in the dataset. The huge amount of data is always suitable for deep learning models. Regarding detection accuracy, our suggested CNN BLSTM hybrid model performs better than the random over-sampling technique, offering over 99%. Tables 6.12, 6.13, and Fig 6.4. above detailed the CNN BLSTM hybrid model's performance for the UNSW-NB15 imbalance dataset and the multiclass NSL-KDD.

V. CONCLUSION

The previous research from the literature reviews shows that while the detection accuracy for rarely occurring attack classes (U2R, R2L) is low, the average model accuracy for normal traffic in the UNSW-NB15 and NSL-KDD is roughly 99%. Regardless of the type of attack, each poses a threat to network machines equally. To provide a comparative analysis, we juxtapose our results with existing findings of 91.12% [51] and 90.83% [49] detection accuracy for NSL-KDD binary, and 99.70% [61], 82.08% [73] 82.08% for UNSW-NB15 binary datasets. Our experiments enhance accuracy to 98.27% on NSL-KDD and 99.87% on UNSW-NB15 binary datasets by carefully selecting hyperparameters and conducting various experiments. We explored the CNN BLSTM hybrid model's hyperparameters (dropout. epochs, batch size, learning rate, and optimizer) to maximize detection accuracy for the binary NSL-KDD and UNSW-NB15.

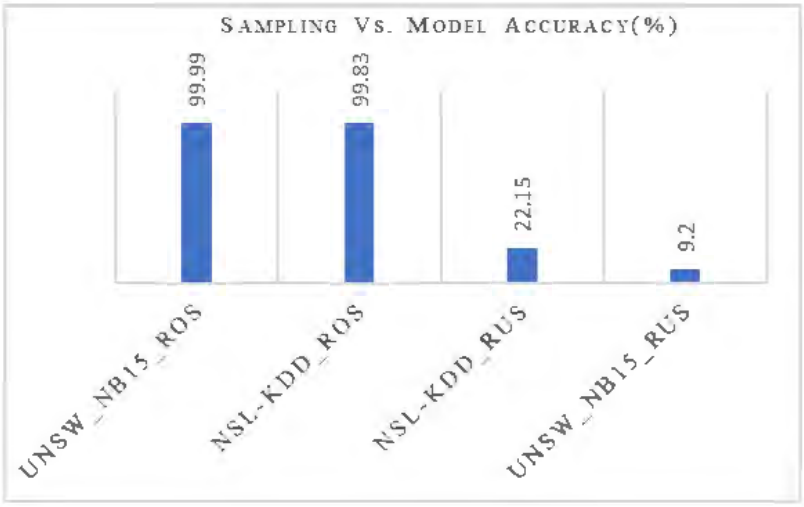

Fig. 6. 4. Sampling vs multi-class accuracy (%) for Bi-LSTM model.

The model performance depends on the combination of hyperparameters, the size of the dataset used to train/test the model, and the selection of the machine learning/ deep learning model. Our research provides information about the data size used during the experiments and the choice of hyperparameters. The suggested model uses random over-sampling techniques on a single set of data to provide 99.99% and 99.83% model accuracy for the multiclass UNSW-NB15 and NSL-KDD datasets, respectively (train and test data merge into a single file before sampling).

Selecting random over-sampling or under-sampling relies on the particulars of the dataset and the issue at hand. To achieve a balance, combining the two methods, a practice known as hybrid sampling may occasionally be necessary. It's crucial to remember that there are more sophisticated methods for dealing with class imbalance, such as SMOTE (Synthetic Minority Over-sampling Technique), which creates synthetic instances for the minority class instead of merely copying real instances. Thoroughly examining those approaches in various network intrusion detection multiclass datasets extends this research

effort. The proper use of hyperparameters of neural networks, size of dataset used to train the model, and sampling methods for CNN BLSTM network anomaly model provide the highest detection accuracy for imbalance network data.

CHAPTER 7. ADDRESSING THE CLASS IMBALANCE PROBLEM IN NETWORK-BASED ANOMALY DETECTION

Abstract- Network-based anomaly detection systems play a crucial role in identifying malicious activities within computer networks. However, the class imbalance problem poses a significant challenge to the effectiveness of these systems, where normal network traffic vastly outweighs anomalous instances, leading to biased models favoring majority classes and overlooking minority anomalies. In this paper, we propose a comprehensive approach to tackle the class imbalance problem in network-based anomaly detection on NSL-KDD and UNSW-NB15 multiclass datasets. Our methodology incorporates various techniques, including random over-sampling (ROS), random under-sampling (RUS), Synthetic Minority Over-sampling Technique (SMOTE), Adaptive Synthetic Sampling (ADASYN), SMOTE combined with Edited Nearest Neighbors (SMOTEENN), and the number of class reduction technique to deal with the imbalance data problem to enhance the detection performance. We empirically evaluate our approach on NSL-KDD and UNSW-NB15 network-based intrusion datasets, demonstrating its effectiveness in improving the bidirectional long-short memory (Bi-LSTM) performance metrics. Our findings indicate that addressing class imbalance significantly enhances the robustness and reliability of network-based anomaly detection systems, thereby contributing to the advancement of cybersecurity measures in modern network environments.

Keywords— ADASYN, Bi-LSTM, class reductions, data imbalances, machine learning, ROS, RUS, SMOTE, SMOTEENN.

I. INTRODUCTION

The data generation trend and advancement of information technology have witnessed a remarkable surge in recent years, characterized by an exponential growth in the volume and variety of data produced globally. This surge is propelled by the widespread adoption of digital technologies such as smartphones, IoT devices, social media platforms, and cloud computing, resulting in an unprecedented proliferation of data sources and formats. Concurrently, advancements in information technology, including the development of big data analytics, machine learning algorithms, and edge computing infrastructure, have facilitated the efficient processing, analysis, and utilization of this vast amount of data. These advancements have not only enabled organizations to derive valuable insights and make data-driven decisions but have also catalyzed innovations across various sectors, from healthcare and finance to transportation and manufacturing, driving societal progress and economic growth on a global scale.

Network anomaly and outlier detection is a crucial aspect of cybersecurity and network management, aimed at identifying abnormal behavior or events within computer networks that deviate from expected patterns. These anomalies can indicate various security threats, including intrusions, malware infections, denial-of-service attacks, and insider threats. Outliers, on the other hand, refer to data points or events that significantly differ from most network traffic or behavior. Anomaly detection techniques in network security encompass various methods, including statistical analysis, machine learning algorithms, and deep learning models, to detect irregularities in network traffic, packet payloads, user access patterns, and system configurations. These techniques analyze network data such as traffic volume, packet headers, protocol usage, and connection patterns to identify suspicious

activities or deviations from normal network behavior. By continuously monitoring and analyzing network traffic, anomaly and outlier detection systems can promptly detect and mitigate cybersecurity threats, safeguarding network infrastructure, data assets, and critical resources from potential breaches or attacks.

Network anomaly detection methods encompass diverse techniques aimed at identifying abnormal behavior or events within computer networks. These methods utilize various approaches, including statistical analysis, machine learning algorithms, and rule-based systems, to detect deviations from expected network behavior. Statistical methods typically involve establishing baseline models of normal network activity and flagging deviations that exceed predefined thresholds. Machine learning algorithms, such as supervised, unsupervised, and semi-supervised techniques, learn patterns from labeled or unlabeled network data to detect anomalies based on features like traffic volume, packet characteristics, or user behavior. Rule-based systems rely on predefined rules or signatures to identify known attack patterns or suspicious activities. Furthermore, advanced anomaly detection approaches leverage anomaly scoring mechanisms, anomaly clustering techniques, and ensemble learning methods to enhance detection accuracy and reduce false positives. By employing these methods and continuously monitoring network traffic, anomaly detection systems can effectively detect and mitigate cybersecurity threats, ensuring the integrity, availability, and confidentiality of network resources and data assets.

Data imbalance poses a significant challenge in network anomaly detection, where the occurrence of normal network behavior significantly outweighs that of anomalous activities. This imbalance in class distribution can lead to biased models prioritizing accuracy on the majority class while neglecting the detection of minority class anomalies.

Imbalanced datasets can hinder the performance of anomaly detection algorithms, resulting in high false negative rates and diminished detection sensitivity for rare anomalies. Traditional machine learning models trained on imbalanced data may struggle to learn meaningful patterns from the minority class, leading to poor generalization and limited anomaly detection capability. Addressing data imbalance in network anomaly detection requires specialized techniques such as resampling methods, for example, oversampling, and under sampling, algorithmic adjustments, for example, cost-sensitive learning and class weighting, ensemble learning approaches and anomaly generation techniques, such as, synthetic data generation. Additionally, adopting advanced anomaly detection methods, such as deep learning architectures and anomaly-specific algorithms, can help mitigate the impact of data imbalance by leveraging the representation learning capabilities of deep neural networks and the flexibility of anomaly-centric models. By addressing data imbalance effectively, network anomaly detection systems can achieve improved detection performance, enhanced resilience against cybersecurity threats, and better overall network security posture.

Performance improvement techniques in network anomaly detection play a pivotal role in enhancing the effectiveness and efficiency of anomaly detection systems. These techniques encompass a variety of strategies aimed at optimizing the detection accuracy, reducing false positives, and enhancing the responsiveness of anomaly detection algorithms. Key approaches include feature selection and engineering, where relevant features from network data are identified and transformed to enhance the discriminatory power of anomaly detection models. Algorithm selection and optimization are equally crucial, as choosing appropriate anomaly detection algorithms and fine-tuning their parameters can

significantly impact detection performance. Ensemble learning methods, such as bagging and boosting, further improve performance by combining multiple anomaly detection algorithms to leverage their complementary strengths and enhance detection accuracy and robustness.

Additionally, data preprocessing techniques, including normalization and outlier removal, improve data quality and consistency, leading to more reliable anomaly detection outcomes. Rigorous evaluation methodologies, such as cross-validation and the use of appropriate evaluation metrics, ensure accurate performance assessment and model validation, guiding further refinement and enhancement efforts. By integrating these performance improvement techniques, network anomaly detection systems can achieve higher detection accuracy, lower false positive rates, faster response times, and greater resilience against cybersecurity threats, thereby fortifying the overall security posture of network infrastructures and safeguarding critical data assets.

Different types of sampling techniques, including under-sampling, oversampling, and hybrid sampling, are commonly employed in network anomaly detection to address challenges associated with imbalanced datasets and enhance the performance of anomaly detection systems. Under-sampling techniques involve reducing the number of instances in the majority class by randomly selecting a subset of data points, thereby balancing class distributions. This approach aims to prevent the dominance of the majority class and improve the model's ability to detect minority class anomalies. Conversely, oversampling methods focus on increasing the representation of the minority class by either replicating existing instances or generating synthetic samples. Techniques like SMOTE (Synthetic Minority Over-sampling Technique) create synthetic instances based on the characteristics

of existing minority class samples, effectively expanding the dataset and alleviating class imbalance. Hybrid sampling techniques combine elements of both under-sampling and oversampling approaches to strike a balance between retaining important information from the majority class and providing sufficient representation for the minority class. By leveraging these sampling techniques, network anomaly detection systems can achieve better generalization, mitigate the impact of data imbalance, and enhance their ability to detect anomalous activities with higher accuracy and reliability in real-world network environments. In supervised machine learning, reducing the number of classes is a technique employed to simplify the classification task and potentially improve model performance, particularly in scenarios with a large number of classes or imbalanced class distributions.

This reduction can be achieved through several methods. Class Aggregation or Merging; similar classes can be merged or aggregated into a single class, reducing the overall number of distinct classes in the dataset. This approach helps simplify the classification problem by grouping related classes together and reducing the complexity of the model.

Class elimination is the technique where some classes may need to be more relevant or contribute minimally to the classification task. Removing these classes from the dataset can reduce noise and focus the model's attention on the most important and discriminative classes, potentially leading to improved performance. Hierarchical classification organizes classes into a hierarchical structure, where higher-level classes represent broader categories, and lower-level classes represent more specific subcategories. Reducing the number of classes at higher levels of the hierarchy makes the classification task more manageable, allowing the model to focus on finer distinctions within fewer categories. Thresholding is

where classes with low frequencies or instances may be combined into a single "other" or "miscellaneous" class if they contribute little to the overall classification performance. This simplifies the problem by reducing the number of distinct classes while capturing important information.

Reducing the number of classes in supervised machine learning can lead to a more focused and efficient classification task, especially when the original class structure is overly complex or noisy. However, it is essential to carefully consider the implications of class reduction on the interpretability and generalization of the model, as well as the potential loss of information resulting from merging or eliminating classes. Evaluating the impact of class reduction techniques on overall model performance through rigorous testing and validation is crucial to ensure that the simplified classification task still effectively captures the underlying patterns and relationships in the data.

II. RELATED WORK

The network anomaly is detected by combining different methods, including data preprocessing, working with different combinations of machine learning algorithms to improve the performance. The data imbalance is one of the huge problems due to the large regular set of data generation compared to the anomaly present in it. The dataset class containing a small number of instances is called the minority class. The traditional classification models perform negatively with the unbalanced dataset called the class imbalance problem. Authors [80] mention that class imbalance problems can be solved using different methods, such as working with data and algorithms and combining both data and algorithms. The oversampling and under-sampling methods are mostly used methods to balance the data. Authors compare various oversampling techniques, including SMOTE,

ADASYN, Borderline-SMOTE, and Safe-Level SMOTE, for different sets of classifiers and performance metrics. Authors [81] researched the class imbalance problem from previously published research works. Data level and algorithm level methods are two techniques for dealing with data imbalance problems. Data level methods mostly focus on sampling the training data to deal with the class imbalance problems. In contrast, the algorithm-level methods depend on the different machine learning or deep learning methods used for modeling the data. The data level approach includes data sampling, feature selection, etc. Similarly, the algorithm-level methods include ensemble methods and hybrid methods.

The authors [82] used data augmentation to deal with the data imbalance problem using different sets of models, including K-means, Random Forests, and Neural Networks. GAN is used to generate adversarial samples based on the attack class used. The data is selected in percentage values from the whole dataset to train and test the model such as 1%, 3%, 50%, and more. During this study, Generative Adversarial Network (GAN) showed a slightly better performance than SMOTE in detecting the anomaly. During this work, the researcher focused on working on data to deal with imbalance problems in network-based anomaly detection.

Authors [83] proposed the hybrid deep learning model that combines CNN and BLSTM models to evaluate the effectiveness of five deep learning models on different network datasets, including CIC-IDS2017, IoT-23, and UNSW-NB15. The overall model accuracy was 76.32% for the multiclass UNSW-NB15 dataset. During this work, the authors worked to find a better hybrid model for the network-based anomaly detection on the imbalance dataset.

Authors [84] used the attention-based LSTM and attention-based bidirectional LSTM intrusion detection with the SMOTE technique to balance the UNSW-NB15 dataset to deal with the data imbalance problem. The authors achieved the highest detection rate of 93%. The authors [85] used reinforcement learning combined with SMOTE techniques to address the data imbalance problems on the NSL-KDD dataset. In this research work, authors compared the performance metrics using techniques including SMOTE, ROS, NearMiss1, and NearMiss2 and found the highest accuracy of 82% and F1 of 82.4%.

Authors [50] used the one-side selection and SMOTE sampling to deal with the data imbalance problem during the data preprocessing stages. Then convolution neural network combined with the bidirectional long short-term memory hybrid model experimented on NSL-KDD and UNSW-NB15 datasets with model accuracy of 83.58% and 77.16%, respectively. Authors [50] implemented the hybrid deep learning models consisting of the combination of CNN and BLSTM models where they used one-side selection to reduce the noise samples from the majority classes and then synthetic minority over-sampling (SMOTE) strategies to deal with the data imbalance for the NSL-KDD and UNSW-NB15 network traffic datasets. Their model provided the highest accuracy of 83.58% for NSL-KDD and 77.16% for the UNSW-NB15 dataset. Authors [84] implemented the attention-based long short-term memory (AT-LSTM) and attention-based bidirectional LSTM (BLSTM) on the UNSW-NB15 dataset. They implemented synthetic minority over-sampling (SMOTE) to balance the dataset and achieved a 93% detection rate. The detection rate of analysis attacks is very low which is 7% for AT-LSTM and 8% for AT-BLSM models while the normal and generic class detection rate is 100%.

Authors [86] proposed the combination of Genetic Algorithms (GA) and GAN to improve anomaly detection for class imbalance problems in their case study-based research work. The multi-objective GAN (MOGAN) generates samples related to the minority classes to address the class imbalance problems. Authors [87] proposed an LSTM-based novel integration of the chaotic butterfly optimization algorithm (CBOA) and particle swarm optimization (PSO) to improve the accuracy of the LSTM algorithm to efficiently detect the network-based intrusion for a binary and multiclass dataset, including NSL-KDD and LITNET-2020. They achieved a model accuracy of 93.09% on the binary KDDTest+ dataset and 86.89% on the binary KDDTest-21 dataset. Authors [87] proposed an improved version of the long short-term memory model, which integrated the chaotic butterfly optimization algorithm (CBOA) and particle swarm optimization (PSO) to improve the detection accuracy on multiclass and binary class NSL-KDD and LITNET2020 datasets. The model provides the highest accuracy of 91.31% on the KDDTest+ dataset. The hybrid sampling methods deal with the data imbalance problem of the NSL-KDD dataset.

Authors [49] proposed the deep learning model for network anomaly detection, which combines an attention mechanism, convolutional neural network, and bidirectional LSTM to detect the network traffic anomaly. They implemented adaptive synthetic sampling (ASASYN) to deal with the class imbalance problem on the NSL-KDD dataset with the highest accuracy of 90.73% and F1-score of 89.65%. Authors [88] implemented the hyperparameter tuning and random synthetic over-sampling techniques in the CNN network combined with the Bi-LSTM model to increase the model efficiency in terms of accuracy and F-measure of 98.36% and 99.05%, respectively, on the NSLKDD dataset.

Authors [89] implemented the CNN with BLSTM model selecting features using a random forest classifier along with the recursive feature elimination approach to evaluate the performance of the binary and multiclass NSL-KDD and UNSW-NB15 datasets. Their model produces the highest accuracy of 95.96% on the binary NSL-KDD dataset and 93.51% accuracy on the binary UNSW-NB15 dataset. They applied random oversampling to deal with the class imbalance problem, which resulted in an average detection accuracy of 98.42% for the NSL-KDD multiclass dataset and 84.23% average accuracy for the UNSW-NB15 multiclass dataset.

Authors [90] purposed the attention-based LSTM model and different dimensionality reduction techniques including Chi-square, UMAP, principal components analysis, and mutual information are used on the NSL-KDD dataset. The 3 component PCA dimensionality reduction provides the highest accuracy of 99.09% on binary and 98.49% on multiclass NSL-KDD datasets. Authors [91] proposed an algorithm based on sampling and improved One-vs-All (OVA) where the dataset is down-sampled with the majority classes during an auxiliary classifier generative adversarial network. The model accuracy was improved using OVA-based model training and testing.

In the previous work [53], we implemented the heterogeneous ensemble methods to improve the performance network intrusion detection system. In [54] , the in-depth comparison of the performance of the binary and multiclass network intrusion system where the number of classes is reduced to enhance the performance. In [78], [76], the hyperparameters are tuned to enhance the network anomaly detection. Similarly, in [92] , [79] the research is based on the sampling methods to deal with the network imbalance dataset using the deep learning models.

From the above literature reviews, the authors [80], [81] performed detailed studies regarding data imbalance problems and different techniques to overcome the issues related to data imbalance. Authors [82] implemented the learning-based data augmentation techniques, which suffer from the model collapse problem. Authors in [83] did not reference the selection of the hyperparameters, such as optimizers and dropout rates. In [84], [85], the data preprocessing where the training and testing dataset contains unequal numbers of data entries, which generate the different numbers of features in training and testing datasets. The information regarding the training and testing datasets is missing. In [50] [84] , authors implemented the SMOTE techniques and hybrid deep learning algorithms to improve the performance. The information on the hyperparameters is not clear in [88] and also the training and testing dataset (70:30 train and test) during sampling information. In [86], the disadvantage of GA is slow convergence, so we need to do hyperparameters tunning the used model. In [87], data preprocessing and model optimization techniques are implemented to enhance the model performance for imbalanced network datasets. The data preprocessing and one hot encoding details information needs to be included in their works.

Authors [49] implemented ADASYN sampling on U2R and R2L of the training dataset. The selection of Network hyperparameters needs to be clarified. In [89] [90] [91], sampling, feature engineering, and model selections were implemented to enhance the performance of network anomaly detection. In [53] [54] [78] [76] [92] [79], The research work focused on data and algorithms without missing data preprocessing and tuning hyperparameters in anomaly detection algorithm.

Further, our research work will fill the gap of the data preprocessing, and hyperparameters selection to enhance the anomaly detection in imbalance dataset. This

research work focused to find the insight of the sampling methods and combinations on training and testing dataset. This work contributes in the field of machine learning and imbalance data analysis.

The main contributions of this research work are:

1) Investigating the effect of different types of random over-sampling, random under-sampling, and hybrid sampling methods on the Bi-LSTM model.
2) Investigating the sampling methods on different combinations of training and testing data.
3) Investing the data preprocessing for training and testing data where the different numbers of categorical values are present such as training data and test data set have different entries of categorical data which generates the unequal numbers of columns during one hot encoding.
4) Investigating the different number of classes for supervised learning-based anomaly detection to enhance the performance without deleting the data entries of the imbalance dataset.
5) The Bi-LSTM-based model produces the best F1-score when we apply the sampling on the single dataset and implement the train test split ratio for the model training and testing dataset. The random over-sampling produces an F1 score of greater than 99.9% (99.91 % for the KDD dataset and 99.93% for the UNSW-NB15 dataset)
6) Investing the sampling Vs. reduction in the number of classes for imbalanced datasets in the field of network anomaly detection.

The remainder of the paper covers three sections. Section II describes the system model of the Bi-LSTM approach. Section III illustrates the results and discussion, while Section IV concludes this research work.

III. SYSTEM MODEL

The overall proposed model encompasses the following steps.

1. Network Intrusion Datasets
2. Data Pre-processing
3. Training and Testing data preparation
4. Bi-LSTM model preparation
5. Bi-LSTM model training
6. Model testing, evaluation, comparison, and decision.

The overall implementation schematic of the bidirectional LSTM-based model is shown in Fig. 7.1. A detailed discussion of the above-stated methods is provided in the subsequent sections.

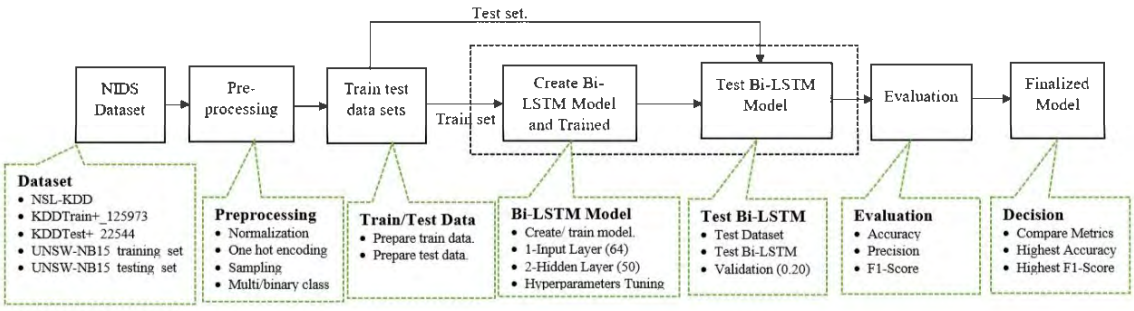

Fig. 7. 1. Block diagram of Bi-LSTM model.

A. Network Intrusion Datasets

This research utilizes two distinct network-based intrusion datasets: NSL-KDD [64] and UNSW-NB15 [75]. Within the NSL-KDD dataset, two subsets are employed for model training and testing purposes. Specifically, KDDTrain+ and KDDTest+ encompass the complete NSL-KDD train and test data, respectively, alongside information regarding difficulty level, target class, and attack types. These subsets comprise 41 features along with details on attack types, difficulty level, and target class. The NSL-KDD subset represents an advancement over the KDD99 dataset, encompassing various attack classes such as Normal, DoS, Probe, R2L, and U2R. KDDTrain+ comprises 125,973 records, while KDDTest+ comprises 22,544 records.

The experiments concerning data imbalance also involve the utilization of UNSW-NB15 datasets. A division of this dataset was designated for training and testing purposes, denoted as UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv, respectively. The training set file contains a total of 82,332 records, while the testing set file contains 175,341 records. The dataset utilized in this research encompasses 43 features, attack categories, and labels. Various types of attacks are represented in this dataset, including Analysis, Backdoor, DoS, Exploit, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms.

B. Data Pre-processing

Since those data are a cleaned version of the KDD99 dataset, there is little work for data preprocessing. The downloaded files are labeled as test and train data. The target class is initially separated from the training and testing datasets to create the class label. Three categorical features, including protocol-type, service, and flag, are extracted from the pool

of feature sets remaining numerical features. The categorical features are converted into numerical values using dummy one-hot encoding methods, while those numerical features are normalized using standard Scalar methods. Both types of feature sets are combined into a single data frame, resulting in final datasets for training and testing datasets.

Since there are different types of services present in the KDDTrain+ dataset and KDDTest+ dataset, the one hot encoding produces unequal numbers of features. The KDDTrain+ dataset contains a total of 126 features, while the KDDTest+ contains a total of 120 features after the implementation of one hot encoding. Those additional features service_aol, service_harvest, service_http_2784, service_http_8001, service_red_i, and service_urh_i are inserted into the KDDTest+ dataset after finding the exact location where those features reside into the KDDTrain+ dataset. We preserved the attacks_types and difficulty_level features because those features are highly relevant to the target class and increase the model's efficiency.

TABLE 7. 1
CLASS INFORMATION ON NSL-KDD

Class	Categories
2	Normal, abnormal
3	Normal, DoS, R2L_probe_U2R
4	Normal, DoS, probe, U2R_R2L
5	Normal, Dos, R2L, probe, U2R

All the attacks in the network or systems help to enter the intruder into the network or system. We used Table 7.1 above for the class reduction experiments to reduce the classes in the KDDTrain+ and KDDTest by combining the different attack classes into the new attack class. For example, class 3 consists of one normal class and two attack classes. Among the attack classes, DoS is the attack class in the given dataset, but the new attack class is the combination of R2L, Probing, and U2R attack classes from the given dataset.

UNSW-NB15 dataset was divided into two sets for training and testing purposes, labeled UNSW_NB15_training-set.csv and UNSW_NB15_testing-set.csv. The UNSW_NB15_training-set comprises 175,341 records, while the UNSW-NB15_testing-set comprises 82,332 records, encompassing various attacks and normal data. Initially, the features on this dataset are 49. Some features such as source IP, destination IP, source port, and destination port are removed from the original dataset before starting the preprocessing. Three features are categorical (proto, service, and state); the remaining are numerical features comprising 43 features and two additional features: attack category and label.

First, those categorical features are converted into numeric using dummy one hot encoding. All numerical features are applied to the standard scalar normalization method. Every time, the target class or label is converted into a numeric using the label encoding method. After preprocessing the numeric and categorical features, 192 features for UNSW_NB15_testing-set data and 196 features for UNSW_NB15_training-set data were generated. Again, here we are taking two sets of data: one we can use for training and the other for testing or vice versa. The categorical values of data entries are not the same for both datasets; hence, the one hot encoding produces unequal numbers of features on both data sets after preprocessing. Some features generated from one hot encoding, such as state_ACC and state_CLO, are not included in the UNSW-NB15_training-set. Similarly, proto_icmp, proto_rtp, state_ECO, state_PAR, state_URN, and state_no features are not included on UNSW_NB15_testing-set. The empty columns are inserted in the exact location of those missing features on the respective dataset, generating 198 features plus one target class.

TABLE 7. 2
CLASS INFORMATION ON UNSW-NB15

Class	UNSW-NB15 traffic categories
2	Normal, Attacks
3	Normal, Generic, Ex_Fu_Do_Re_An_Ba_sh_wo
4	Normal, Generic, Exploits, Fu_Do_Re_An_Ba_sh_wo
5	Normal, Generic, Exploit, Fuzzers, Do_Re_An_Ba_sh_wo
6	Normal, Generic, Exploit, Fuzzers, DoS, Re_An_Ba_sh_wo
7	Normal, Generic, Exploit, Fuzzers, DoS, Recon, An_Ba_sh_wo
8	Normal, Generic, Exploit, Fuzzers, DoS, Recon, Analysis, Ba_sh_wo
9	Normal, Generic, Exploit, Fuzzers, DoS, Recon, Analysis, Backdoor, sh_wo
10	Normal, Generic, Exploit, Fuzzers, DoS, Recon, Analysis, Backdoor, shellcode, worms
Ex-Exploits, Fu-Fuzzers, Do-DoS, Re-Reconnaissance, An-Analysis, Ba-Backdoor, sh-Shellcode,wo-Worms	

Some attack classes consist of small numbers of data, so all the network attacks are equally contributed to letting the intruder in the system or networks. Here, we combined some attacks into new attack categories to reduce the number of classes in the training and testing dataset. Table 7.2 was referenced to create the different numbers of multiclass training and testing datasets. The combination of the different classes was based on the number of attacks present in the class. The regrouping minor attacks created the new types of attack class.

C. Training and Testing Datasets Preparation

The train-test split methodology evaluates the performance of machine learning algorithms in predicting outcomes from unseen data. We opt for a 70:30 split ratio for our Bi-LSTM model when dealing with single-file data preprocessing. However, when using two separate files for training and testing, the specifics regarding the quantity of training and testing data are elucidated in the preceding data preprocessing section above.

D. Bi-LSTM Model Preparation

A recurrent neural network (RNN) utilizes feedback loops to analyze sequential data and make predictions. RNNs store past and future state information in their memory, making them effective for tasks like speech recognition, language processing, and image classification. Long Short-Term Memory (LSTM) networks address the issue of vanishing gradients in RNNs by employing memory blocks and three multiplicative units known as input, output, and forget gates. These gates, akin to write, read, and reset operations, allow LSTM cells to retain and retrieve data over prolonged periods, mitigating the vanishing gradient problem. Bidirectional RNNs merge two separate RNNs to process input in both forward and backward directions. The forward and backward LSTM networks form the two components of a Bidirectional LSTM (Bi-LSTM). The forward LSTM extracts feature in the forward direction, while the backward LSTM does so in the reverse. By utilizing sequences from both past and future contexts, the Bi-LSTM predicts or labels each element in a sequence. This is achieved through two LSTMs operating sequentially, one from left to right and the other from right to left.

E. Bi-LSTM model training

The architecture of the Bi-LSTM model's neural network is prepared for training. The datasets are structured to have separate sets for training and testing, or vice versa. The split percentage determines the allocation of data for training and testing when dealing with a single dataset. The selection of model training hyperparameters is conducted through various experiments, refining parameters such as epochs and batch size to enhance detection efficiency. Additionally, 20% of the training data is set aside for validation purposes in assessing the Bi-LSTM model's performance.

F. Model testing Evaluation, Comparing and Making decisions

The model is constructed and trained using the training dataset with specified hyperparameter values. The training duration varies based on these parameters. Following training, the model evaluates its performance on unseen data. As there are no predefined rules for hyperparameter selection, a combination of random selection and fine-tuning is employed through different experiments. Post-testing, performance metrics are determined, and tailored to the specific machine learning model used. In supervised learning, ground truth values are utilized to assess metrics such as detection accuracy, precision, F1-Score, Recall, program execution time, Area under the ROC curve, etc. Keras' classification report generates metrics like true positives, true negatives, false positives, and false negatives. The confusion matrix furnishes comprehensive performance evaluation for the given training and testing datasets.

Various experiments are conducted to find optimal hyperparameter values for improved results. These experiments involve comparing the accuracy and F1-score across different configurations of the optimizer, epochs, batch size, and train-test split ratio for the Bi-LSTM model. Additionally, our model's performance parameters are compared with existing research results to gauge its effectiveness. Specifically, we recorded and compared F1 scores for NSL-KDD and UNSW-NB15 multiclass NIDS datasets.

IV. RESULTS AND DISCUSSIONS

The intrusion detection system utilizes machine learning and deep learning techniques to identify anomalies. Python scripting is employed to develop code for implementing a network intrusion detection model, leveraging packages such as NumPy, Pandas, Keras, and sci-kit-learn. Additionally, tools like WEKA, Java, C#, Visual C++, and

MATLAB are commonly used for intrusion detection systems. To ensure reproducibility, seed values are set to yield consistent results across multiple runs on the Jupyter Notebook platform. Experimental findings are analyzed using plots or tables within the MS Office suite. These experiments are conducted on a Windows machine with 16GB RAM and an i7 processor. The versions of Python and its packages used in this research are Python 3.7.12, Keras 2.6.0, and TensorFlow 2.9.1. Various experiments are conducted to compare performance across different class sizes and sampling methods for the Bi-LSTM model using NSL-KDD and UNSW-NB15 datasets, detailed in subsequent sections.

Under-sampling is a simple yet effective method for dealing with class imbalance in datasets. It entails retaining all data from the minority class while decreasing the volume of data in the majority class. This technique is valuable for data scientists seeking to improve the accuracy of insights derived from imbalanced datasets. During under-sampling, random samples from the majority class are removed until a balanced distribution is attained. While reducing data volume can mitigate storage limitations and improve processing efficiency, it's essential to acknowledge that this process may lead to the loss of valuable information.

The over-sampling technique is used to address dataset scarcity by increasing the number of rare samples, and counteracting imbalance. Unlike under-sampling, where excess samples are discarded, oversampling methods create new rare samples. Techniques like replication, bootstrapping, and SMOTE (Synthetic Minority Over-Sampling Technique) are employed for this purpose. SMOTE involves generating synthetic data points for the minority class by selecting k nearest neighbors to determine the level of oversampling.

Random over-sampling generates the new samples randomly with the replacement of current available samples while SMOTE generates the synthetic data without

replacement. The ADASYN produces a different number of samples by estimating the local distribution of the class to be oversampled. The SMOTEENN is a hybrid sampling method that works by combining SMOTE and Edited nearest neighbors (ENN) methods to generate the samples. While the random under-sampling removed the samples from the majority class with or without replacement. The sampling is implemented in Experiment 1 and Experiment 2.

A. Experiment-1 Sampling Vs. Bi-LSTM Model Performance on NSL-KDD dataset.

Four different sets of experiments were performed on the NSL-KDD dataset. The oversampled techniques add the data samples to the minority class hence the samples on all the classes become equal. The under-sampling removed the data samples from the majority class and the number of samples in the majority class was equal with the minority class. The sampling techniques either delete the samples or add redundancy into the dataset implementing the methods used in the datasets.

TABLE 7. 3
SAMPLING ON TRAINING DATASET AND F1_SCORE ON NSL-KDD

Sampling on the training data set, model test using the original testing dataset, F1_score (%) on NSL-KDD					
Epochs= 50, Batch_size= 512, Training_data = KDDTrain+(sampling), Testing_data = KDDTest+					
Class	**F1_ru**	**F1_ro**	**F1_smt**	**F1_adsyn**	**F1_smteen**
DoS	79.74	89.75	89.91	91.04	92.25
Probe	75.29	77.53	75.73	76.7	76.19
U2R	24.1	16.67	33.53	36.19	21.88
R2L	67.16	56.34	56.9	67.25	62.34
Normal	87.47	97.23	97.07	95.42	99.66
Wt_Avg	80.82	87.17	87.08	88.18	**89.68**
Macro_Avg	66.75	67.5	70.63	**73.32**	70.46
ru: Radom under-sampling, ro: Random Over-sampling, smt: SMOTE, adsyn: ADASYN, smteen: SMOTEENN, F1:F1_score (%)					

Hyperparameters for the Bi-LSTM models were chosen based on prior research [79] . Initially, various sampling techniques were applied to KDDTrain+ for training the

Bi-LSTM model. Subsequently, the pre-processed version of KDDTest+ was used to evaluate the model, and F1 scores were recorded in Table 7.3 for different sampling methods. The Bi-LSTM model was trained for 50 epochs with a batch size of 512. F1 scores were recorded using both weighted and macro averaging, which are suitable metrics for evaluating performance on imbalanced multiclass datasets. Among the sampling methods, the Bi-LSTM model achieved the highest performance with an F1 score of 89.68% using hybrid sampling methods (SMOTEENN) in terms of weighted average, while ADASYN yielded the best result in terms of macro-averaged F1 score at 73.32%, outperforming random oversampling, SMOTE, and random under sampling.

TABLE 7. 4
SAMPLING INDIVIDUALLY ON TRAINING AND TESTING DATA AND F1-SCORE FOR NSL-KDD

Sampling individually on both training and testing dataset, F1-Score (%) on NSL-KDD					
Epochs= 50, Batch_size= 512, Training_data = KDDTrain+(sampling), Testing_data = KDDTest+(sampling)					
Class	**F1_ru**	**F1_ro**	**F1_smt**	**F1_adsyn**	**F1_smteen**
DoS	74.42	88.15	78.58	81.31	81.83
Probe	84.85	73.42	77.65	73.54	78.64
U2R	75.81	70.99	56.39	26.24	58.74
R2L	61.67	36.72	42.87	44.23	47.6
Normal	77.58	90.85	92.82	92.84	99.15
Wt_Avg	74.86	**78.07**	69.66	63.63	73.19
Macro_Avg	**74.86**	72.02	69.66	63.63	73.2
ru: Radom under-sampling, ro: Random Over-sampling, smt: SMOTE, adsyn: ADASYN, smteen: SMOTEENN, F1:F1_score (%)					

Table 7.4 displays the performance metrics of the Bi-LSTM model when both the training (KDDTrain+) and testing data (KDDTest+) are individually sampled. Each dataset underwent separate preprocessing procedures, resulting in smaller datasets for under-sampling and larger ones for oversampling. Following sampling, both training and testing datasets achieved balance. The F1 scores exhibited different outcomes compared to Table 7.3, indicating improved balance during model testing. The model achieved its highest

weighted average F1 score of 78.07% with naive random oversampling and its highest macro average of 74.86% with naive random under-sampling methods.

TABLE 7. 5
SAMPLING ON THE COMBINED DATA (TRAINING AND TESTING) AND F1-SCORE ON NSL-KDD

Sampling on combined data set (training and testing), train-test split, F1_score (%) on NSL-KDD					
Epochs= 50, Batch_size= 512, Data = combine (KDDTrain+KDDTest+) (sampling)					
Class	**F1_ru**	**F1_ro**	**F1_smt**	**F1_adsyn**	**F1_smteen**
DoS	97.06	99.96	99.98	99.97	99.99
Probe	98.41	99.67	99.95	99.96	99.93
U2R	87.8	99.94	99.56	99.48	99.68
R2L	88.52	98.16	99.54	99.42	99.62
Normal	98.46	99.99	99.99	99.99	1
Wt_Avg	94.69	**99.91**	99.81	99.77	99.85
Macro_Avg	94.05	99.54	99.81	99.76	**99.84**
ru: Radom under-sampling, ro: Random Over-sampling, smt: SMOTE, adsyn: ADASYN, smteen: SMOTEENN, F1:F1_score (%)					

The train-test split method was employed to separate training and testing data after applying sampling techniques to a unified dataset. Table 7.5 and Table 7.6 present the performance of the Bi-LSTM model when data were sampled into a single file format. Both KDDTrain+ and KDDTest+ data were merged into a single dataset and pre-processed to create training and testing data. The Bi-LSTM model achieved its highest performance, as shown in Table 7.5, with oversampling methods. It attained a weighted F1-Score of 99.91% with random oversampling and a macro average of 99.84% with hybrid sampling. Similarly, using the single dataset named KDDTrain+ for preprocessing, followed by train-test splitting, the Bi-LSTM model yielded comparable results, as shown in Table 7.6, with a weighted average F1-score of 99.97% during random oversampling and a macro average of 99.84% during hybrid sampling. Thus, employing the single dataset for preprocessing

before applying the train-test split method consistently produced the best results for the Bi-LSTM model.

TABLE 7. 6
SAMPLING ON SINGLE DATASET AND F1-SCORE ON NSL-KDD

Sampling on single data set, train-test split, F1_score (%) for NSL-KDD					
Epochs= 50, Batch_size= 512, Data = KDDTrain+(sampling)_train_test_split					
Class	**F1_ru**	**F1_ro**	**F1_smt**	**F1_adsyn**	**F1_smteen**
DoS	1	99.98	99.99	99.99	99.99
Probe	93.33	99.91	99.97	1	1
U2R	66.67	99.97	99.66	99.12	99.79
R2L	84.21	98.09	99.65	99.09	99.79
Normal	1	99.99	1	1	1
Wt_Avg	90.39	**99.97**	99.85	99.64	99.91
Macro_Avg	88.84	99.59	99.85	99.64	**99.91**
ru: Radom under-sampling, ro: Random Over-sampling, smt: SMOTE, adsyn: ADASYN, smteen: SMOTEENN, F1:F1_score (%)					

B. Experiment-2 Sampling Vs. Bi-LSTM Model Performance on UNSW-NB15 dataset.

Experiments were conducted on the multiclass UNSW-NB15 dataset. Table 7.7 displays performance metrics obtained by sampling the training set using various techniques during preprocessing. The Bi-LSTM model underwent training for 50 epochs with a batch size of 512, consistent with prior experiments. Subsequently, the model was tested on the unaltered testing dataset of UNSW-NB15.

TABLE 7. 7
SAMPLING ON TRAINING DATA AND F1-SCORE ON UNSW-NB15

Sampling on training data set, model test using original testing dataset, F1_score (%) on UNSW-NB					
Epochs=50, Batch_size=512, Training_data = training-set_175341 (sampling), Testing_data = testing-set_82332					
Class	**F1_ru**	**F1_ro**	**F1_smt**	**F1_adsyn**	**F1_smteen**
Analysis	64.38	71.75	71.04	71.45	70.63
Backdoor	4.25	2.46	1.75	1.9	1.75
DoS	52.08	8.59	17.1	6.05	5.63
Exploits	75.12	44.62	68.17	59.55	34.77
Fuzzers	80.36	39.07	85.43	67.6	75.96
Generic	79.74	83.62	81.66	64.61	96.15

Normal	68.34	89.53	87.31	74.73	95.83
Reconn.	26.06	99.63	99.83	99.3	81.5
Shellcode	0	98.31	98.44	97.8	98.57
Worms	0	70.59	85.71	72.46	87.18
Wt_Avg	69.32	74.07	79.64	67.03	**80.23**
Macro_Avg	45.03	60.82	**69.64**	61.55	64.8
ru: Radom under-sampling, ro: Random Over-sampling, smt: SMOTE, adsyn: ADASYN, smteen: SMOTEENN, F1:F1_score (%)					

Results in Table 7.7 indicate that hybrid methods yielded the highest weighted average F1-score of 80.23%, while SMOTE sampling achieved the highest macro average F1-score of 69.64%. Although different sampling methods balanced the training dataset, the testing dataset remained imbalanced after preprocessing.

TABLE 7. 8
SAMPLING INDIVIDUALLY ON BOTH TRAINING/TESTING SET AND F1-SCORE ON UNSW-NB15

Sampling individually on both training and testing dataset, F1_score (%) on UNSW-NB					
Epochs=50, Batch_size=512, Train_set = training-set_175341 (sampling), Test_set= testing-set_82332(sampling)					
Class	**F1_ru**	**F1_ro**	**F1_smt**	**F1_adysn**	**F1_smteen**
Analysis	70.4	68.69	67.99	68.66	68.74
Backdoor	26.83	9.06	6.28	8.65	10.33
DoS	34.29	13.84	23.41	17.5	29.24
Exploits	75.61	37.58	63.23	44.49	36.43
Fuzzers	90.24	39.98	84.84	81.29	91.61
Generic	91.49	90.45	89.55	71.31	94.34
Normal	62.07	89.21	87.15	72.55	94.08
Recon.	67.35	91.65	99.97	97.36	99.87
Shellcode	75.86	88.1	88.09	79.92	88.39
Worms	0	71.15	84.38	64.78	84.96
Wt_Avg	59.41	59.97	69.49	60.65	**69.8**
Macro_Avg	59.41	59.97	69.49	60.64	**69.8**
ru: Radom under-sampling, ro: Random Over-sampling, smt: SMOTE, adsyn: ADASYN, smteen: SMOTEENN, F1:F1_score (%)					

Table 7.8 presents the performance metrics of the model, where both the training and testing datasets of UNSW-NB15 were individually sampled using various methods. Following data preprocessing, both datasets were balanced. The Bi-LSTM model exhibited strong performance with hybrid sampling (SMOTEENN), achieving an F1-score of

69.80%, outperforming other sampling methods. Notably, the F1-score for the "worms" class was null during under-sampling, as under-sampling randomly deletes samples from the dataset. Due to the small data entries in the minority class in UNSW-NB15, the training data was insufficient for the Bi-LSTM models.

TABLE 7. 9
SAMPLING ON THE COMBINED DATA SET (TRAINING AND TESTING) AND F1-SCORE ON UNSW-NB15

Sampling on combined data set (training and testing), train-test split, F1_score (%) on UNSW-NB					
Epochs=50, Batch_size=512, data = combine(training-set_175341+testing-set_82332) sampling					
Class	**F1_ru**	**F1_ro**	**F1_smt**	**F1_adysn**	**F1_smteen**
Analysis	87.76	99.17	99.99	1	99.92
Backdoor	72.73	98.68	99.98	1	99.91
DoS	77.27	99.61	99.74	99.58	99.87
Exploits	87.06	99.85	99.75	99.59	99.86
Fuzzers	93.67	99.92	99.97	99.98	99.97
Generic	96.63	99.97	99.97	99.99	99.99
Normal	91.57	1	99.97	1	99.98
Recon.	93.67	99.99	99.98	1	99.99
Shellcode	92.5	1	99.99	1	1
Worms	95.05	1	1	1	1
Wt_Avg	88.69	99.93	99.93	99.91	**99.95**
Macro_Avg	88.79	99.72	99.93	99.91	**99.95**
ru: Radom under-sampling, ro: Random Over-sampling, smt: SMOTE, adsyn: ADASYN, smteen: SMOTEENN, F1:F1_score (%)					

In experiment Table 7.9, the training and testing data were merged into a unified dataset and sampled to create a balanced dataset. Likewise, for experiment Table 7.10, the training dataset from UNSW-NB15 underwent preprocessing. Subsequently, trained and test datasets were obtained using a train-test split, and the Bi-LSTM model was trained and tested for both experiments [3a] and [3b]. Notably, the Bi-LSTM model achieved outstanding performance with an F1-score of 99.95% on the UNSW-NB15 dataset.

TABLE 7. 10
SAMPLING ON THE SINGLE DATA SET AND F1-SCORE ON UNSW-NB15

Sampling on single data set, train-test split, F1_score (%) for UNSW-NB
Epochs=50, Batch_size=512, data = training- set_175341(sampling)_ train_ test_split

Class	F1_ru	F1_ro	F1_smt	F1_adsyn	F1_smteen
Analysis	76.36	99.76	99.99	1	99.85
Backdoor	70.83	99.53	99.99	1	99.85
DoS	54.55	99.69	99.87	99.56	99.84
Exploits	62.75	99.88	99.84	99.49	99.81
Fuzzers	89.55	99.97	99.95	1	99.97
Generic	96.55	99.97	99.97	99.93	1
Normal	85.11	99.99	99.95	1	1
Recon.	70.27	1	99.92	1	1
Shellcode	23.53	99.78	1	1	1
Worms	95.83	1	1	1	1
wt._Avg	75.6	**99.95**	**99.95**	99.9	99.93
Macro_Avg	72.53	99.86	**99.95**	99.9	99.93
ru: Radom under-sampling, ro: Random Over-sampling, smt: SMOTE, adsyn: ADASYN, smteen: SMOTEENN, F1:F1_score (%)					

The individual sampling was applied to a single dataset, but this approach did not yield favorable results for the Bi-LSTM model. However, superior performance was achieved when sampling was applied directly to single files, as evidenced by Table 7.9 and Table 7.10.

C. Experiment-3 Class Size Vs. Bi-LSTM Model Performance

In experiments 1 and 2, various sampling techniques were applied during data preprocessing for the imbalanced NSL-KDD and UNSW-NB15 datasets. Random sampling played a crucial role in addressing the data imbalance issue. By working with the number of classes, the data imbalance problem was mitigated. This involved consolidating minority classes into new categories, as outlined in Table 7.1 and Table 7.2, effectively reducing the total number of classes.

TABLE 7. 11
CLASS SIZE VS BI-LSTM PERFORMANCE (TRAINING82332)

Class Size Vs. Bi-LSTM Performance on UNSW-NB15				
Neural Network 64_50_50, Epochs=50, Batch=512, Optimizer=Adam, default activation function, Training_data =testing-set_82332, Testing_data = training-set_175341				
Class	**Accuracy %**	**wt_Recall %**	**wt_F1score %**	**Exe_time (sec)**

2	**99.88**	**99.88**	**99.88**	270.47
8	98.49	98.49	98	261.9
9	92.01	92.01	89.34	265.05
10	91.01	91.05	88.37	269.07

Tables 7.11 and 7.12 display the performance metrics of the Bi-LSTM model on the reduced number of classes in the UNSW-NB15 dataset. In both experiments, the training and testing datasets were swapped to assess model performance. Combining minority classes to create new categories resulted in improved performance. The Bi-LSTM model achieved an accuracy of 99.88% when the larger dataset (training set) was used for testing, and 95.9% when the smaller dataset (testing set) was used for testing.

TABLE 7. 12
CLASS SIZE VS BI-LSTM PERFORMANCE ON UNSW-NB15(TRAINING175341)

Class Size Vs. Bi-LSTM Performance on UNSW-NB15				
Neural Network 64_50_50, Epochs=50, Batch=512, Optimizer=Adam, default activation function, Training_data = training-set_175341, Testing_data = testing-set_82332				
Class	**Accuracy %**	**wt_Recall %**	**wt_F1score %**	**Exe_time (sec)**
2	**95.9**	**95.9**	**95.87**	409.97
8	86.52	86.52	86.79	413.89
9	89.88	89.88	90.25	445.99
10	80.32	80.32	82.55	405.13

By combining minority classes into new categories based on Table 7.1, the number of classes was reduced in the NSL-KDD dataset, resulting in improved performance shown in Table 7.13. The Bi-LSTM model achieved a performance metric of 96.4%. This reduction in the number of classes enhanced data quality by altering the data distribution in the multiclass dataset, consequently boosting model performance.

TABLE 7. 13
CLASS SIZE VS BI-LSTM PERFORMANCE ON NSL-KDD

Class Size Vs. Bi-LSTM Performance on NSL-KDD				
Optimizer= Nadam(lr=0.041), Epochs=50, Batch_size= 512, Training_data = KDDTrain+, Testing_data = KDDTest+				
Class	**Accuracy %**	**wt_Recall %**	**wt_F1score %**	**Exe_time (sec)**
2	**96.39**	**96.39**	**96.4**	**174.35**
3	94.09	94.09	93.98	168.4
4	91.17	91.17	90.53	179.04

5	89.44	89.44	89.02	189.2

V. CONCLUSION

In conclusion, this study has addressed the challenge of class imbalance in network-based anomaly detection using NSL-KDD and UNSW-NB15 datasets. The class imbalance problem presents a significant obstacle in accurately identifying anomalous activities within computer networks, where the abundance of normal traffic overwhelms the instances of anomalies. The literature review shows that the sampling techniques are used to deal with the data imbalance problems in anomaly detection. The above experiments [1-3] show that the over-sampling (ROS, SMOTE, ADSYN) and hybrid over-sampling (SMOTEENN) outperformed the Bi-LSTM model with the highest F1-score of 99.95% on UNSB-SW15 while 99.97% on NSL-KDD as compared result with literature reviews sections. Our results conclude that the model outperformed only when the oversampling was implemented on a single dataset during the preprocessing stages. The sampling implemented in individually into training and testing two different datasets, needs more computation in feature engineering because the present of categorical values of data in the training testing datasets are not always equal which generates the different numbers of features during one hot encoding. Hence, the reduction of the class by creating the new types of class in the dataset is always the best way to deal with data imbalance problems without deleting or adding the random samples into the dataset rather than distributing the minority data into the new class to detect the anomaly in the network anomaly.

CONCLUSION

The application of artificial intelligence (AI) techniques for the enhancement of network anomaly detection has shown remarkable promise in addressing the ever-evolving challenges posed by malicious activities and network vulnerabilities. Throughout this study, we explored various AI methodologies, including machine learning algorithms, deep learning architectures, ensemble techniques, sampling for various combinations of train-test datasets, and class reduction methods to deal with class imbalance network anomaly detection dataset and their effectiveness in detecting and mitigating network anomalies.

In Chapter 1, we explored the heterogeneous ensemble network anomaly detection methods that represented a significant advancement in the field of cybersecurity. Throughout this study, we investigated the effectiveness of combining diverse traditional machine learning algorithms for the purpose of detecting and mitigating network anomalies. Our findings demonstrated that heterogeneous ensemble methods offer several advantages over single-model approaches, including improved detection accuracy and robustness to varying types of anomalies on network-based anomaly detection.

In Chapter 2, the exploration of class reduction methods for imbalance Network Intrusion Detection datasets KDD99, UNSW-NB15, CICIDS2017, and NSL-KDD offered a promising avenue for addressing the challenges posed by skewed class distributions in network traffic data. The reduction of the class changed the distribution of the data in the different classes; hence, the skewed class problem was addressed to increase the anomaly detection performance. By addressing the challenges posed by class imbalance, these techniques enable NIDS to fulfill better their crucial role in safeguarding network

infrastructure, detecting intrusions, and preserving the confidentiality, integrity, and availability of critical assets and data.

In Chapters 3 and 4, we explored hyperparameter tuning in network anomaly detection using deep learning techniques like Bi-LSTM and CNN-BLSTM, which is vital for enhancing the efficacy and robustness of detection systems in complex network environments. The study highlighted the significance of optimizing various hyperparameters, such as learning rates, batch sizes, network architectures, optimizers, and regularization techniques, to improve model performance. By focusing on these factors, we aimed to enhance the ability of deep learning models to detect and mitigate network anomalies effectively.

In Chapters 5, 6, and 7, we examined different sampling techniques for network anomaly detection, highlighting the importance of addressing class imbalance in network traffic data to improve detection system performance. The study explored various methods, including under-sampling, oversampling, hybrid approaches, and ensemble techniques, and their effectiveness in mitigating the challenges posed by imbalanced datasets. By focusing on these techniques, we aimed to enhance the overall effectiveness of anomaly detection systems.

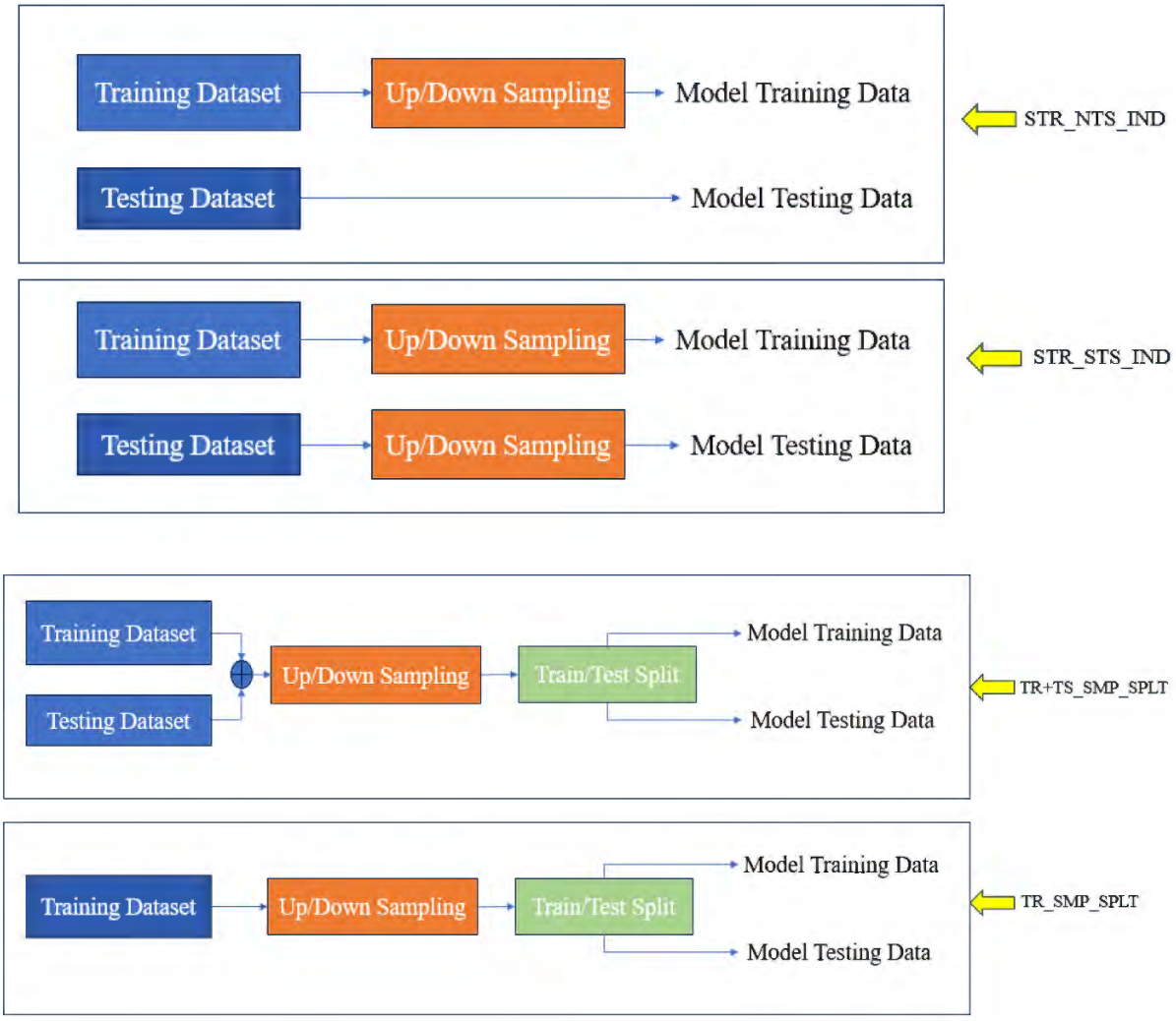

Fig. 8. 1 Combination of the train test data and sampling methods.

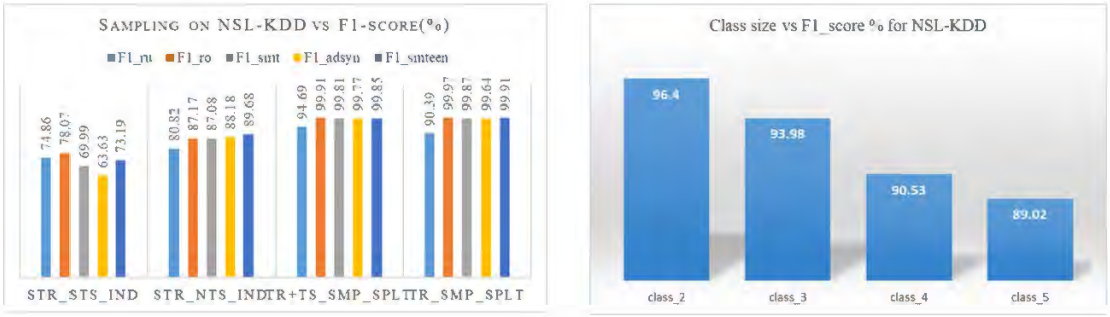

Fig. 8. 2 Comparison of performance for sampling vs class reduction method for NSL-KDD.

The machine learning or deep learning model requires the dataset to be trained and tested for the model evaluation. The training and testing datasets might not always be in separate files like the training data set and testing dataset files. Based on the availability of the given dataset, the various sampling methods play a vital role in determining the efficacy of the model we used. The in-depth exploration of the combination of sampling methods during the training and testing model is shown in Fig 8.1.

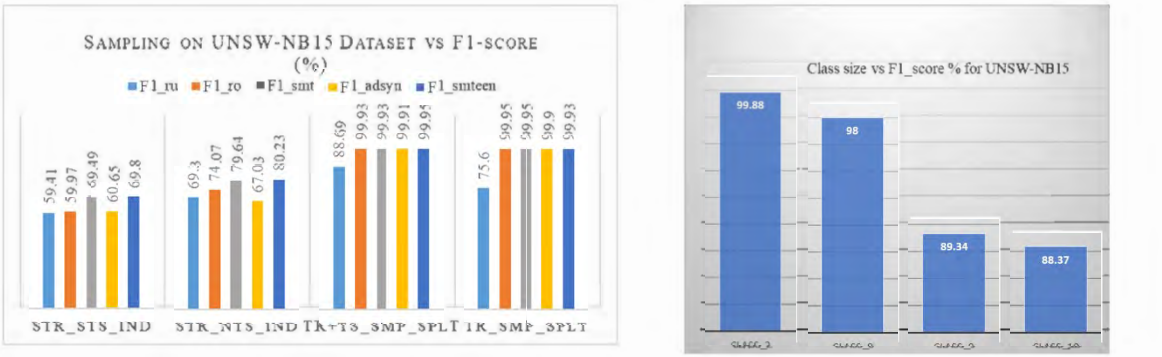

Fig. 8. 3 Comparison of performance for sampling vs class reduction method for UNSW-NB15

The conclusive results are shown in Fig. 8.2 and 8.3, where sampling methods provide better performance on and only the time researcher has a single file data split into train and test purposes after preprocessing. For the real scenario, the training and testing dataset is a separate file format where preprocessing is done separately. The feature selection methods produce different sets of features even if there are the same methods used for those training and testing datasets. The one hot encoding method produces a different set of features on it. Hence, the practicality of dealing with the class imbalance problem using the sampling methods is not as efficient as compared with the class reduction methods. The class reduction methods only change the data distribution by

aggregating the minor classes into new classes, which neither delete nor generate random data.

In conclusion, performance anomaly detection can be increased by dealing with the selection of a model, hyperparameter selection, proper data preprocessing, and reducing the class size to deal with the class imbalance problem. The implementation of our models in the real dataset, working with adversarial attacks, and implementation of the clusters for transfer learning for model training and detect the network-based anomaly are future extensions of this work.

REFERENCES

[1] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity,* vol. 2, p. 1–22, 2019.

[2] H. Wang, J. Gu and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *Knowledge-Based Systems,* vol. 136, pp. 130-139, 2017.

[3] F. Kuang, W. Xu and S. Zhang, "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Applied Soft Computing,* vol. 18, pp. 178-184, 2014.

[4] A. A. Aburomman and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Applied Soft Computing,* vol. 38, pp. 360-372, 2016.

[5] S. Teng, N. Wu, H. Zhu, L. Teng and W. Zhang, "SVM-DT-based adaptive and collaborative intrusion detection," *IEEE/CAA Journal of Automatica Sinica,* vol. 5, pp. 108-118, 2017.

[6] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science,* vol. 89, pp. 213-217, 2016.

[7] B. A. Tama and S. Lim, "Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation," *Computer Science Review,* vol. 39, p. 100357, 2021.

[8] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely and M. M. Fahmy, "A hybrid network intrusion detection framework based on random forests and weighted k-means," *Ain Shams Engineering Journal,* vol. 4, pp. 753-762, 2013.

[9] E. Kabir, J. Hu, H. Wang and G. Zhuo, "A novel statistical technique for intrusion detection systems," *Future Generation Computer Systems,* vol. 79, pp. 303-318, 2018.

[10] P. Tao, Z. Sun and Z. Sun, "An improved intrusion detection algorithm based on GA and SVM," *Ieee Access,* vol. 6, pp. 13624-13631, 2018.

[11] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot," *Sensors,* vol. 17, p. 1967, 2017.

[12] F. Jiang, Y. Fu, B. B. Gupta, Y. Liang, S. Rho, F. Lou, F. Meng and Z. Tian, "Deep learning based multi-channel intelligent attack detection for data security," *IEEE transactions on Sustainable Computing,* vol. 5, pp. 204-212, 2018.

[13] L. Khalvati, M. Keshtgary and N. Rikhtegar, "Intrusion detection based on a novel hybrid learning approach," *Journal of AI and data mining,* vol. 6, pp. 157-162, 2018.

[14] M. A. Abdullah, B. M. Alsolami, H. M. Alyahya and M. H. Alotibi, "Retracted: Intrusion detection of DoS attacks in WSNs using classification techniuqes," *Journal of fundamental and Applied Sciences,* vol. 10, pp. 298-303, 2018.

[15] Y. Li, J.-L. Wang, Z.-H. Tian, T.-B. Lu and C. Young, "Building lightweight intrusion detection system using wrapper-based feature selection mechanisms," *Computers & Security,* vol. 28, pp. 466-475, 2009.

[16] A. K. Shrivas and A. K. Dewangan, "An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD data set," *International Journal of Computer Applications,* vol. 99, pp. 8-13, 2014.

[17] M. A. Ambusaidi, X. He, P. Nanda and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE transactions on computers,* vol. 65, pp. 2986-2998, 2016.

[18] M. Al-Qatf, Y. Lasheng, M. Al-Habib and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access,* vol. 6, pp. 52843-52856, 2018.

[19] A. N. Jaber, M. F. Zolkipli, H. A. Shakir and M. R. Jassim, "Host based intrusion detection and prevention model against DDoS attack in cloud computing," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, 2017.

[20] M. Latah and L. Toker, "An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks," *CCF Transactions on Networking,* vol. 3, pp. 261-271, 2020.

[21] J. Zhao, S. Shetty, J. W. Pan, C. Kamhoua and K. Kwiat, "Transfer learning for detecting unknown network attacks," *EURASIP Journal on Information Security,* vol. 2019, pp. 1-13, 2019.

[22] X. Gao, C. Shan, C. Hu, Z. Niu and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access,* vol. 7, pp. 82512-82521, 2019.

[23] Z. Yueai and C. Junjie, "Application of unbalanced data approach to network intrusion detection," in *2009 First International Workshop on Database Technology and Applications*, 2009.

[24] M. R. Parsaei, S. M. Rostami and R. Javidan, "A hybrid data mining approach for intrusion detection on imbalanced NSL-KDD dataset," *International Journal of Advanced Computer Science and Applications,* vol. 7, pp. 20-25, 2016.

[25] N. Qazi and K. Raza, "Effect of feature selection, SMOTE and under sampling on class imbalance classification," in *2012 UKSim 14th International Conference on Computer Modelling and Simulation*, 2012.

[26] A. I. Al-issa, M. Al-Akhras, M. S. ALsahli and M. Alawairdhi, "Using machine learning to detect DoS attacks in wireless sensor networks," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019.

[27] S. S. Dhaliwal, A.-A. Nahid and R. Abbas, "Effective intrusion detection system using XGBoost," *Information,* vol. 9, p. 149, 2018.

[28] M. A. Khan and Y. Kim, "Deep Learning-Based Hybrid Intelligent Intrusion Detection System," *CMC-COMPUTERS MATERIALS & CONTINUA,* vol. 68, pp. 671-687, 2021.

[29] F. Folino, G. Folino, M. Guarascio, F. S. Pisani and L. Pontieri, "On learning effective ensembles of deep neural networks for intrusion detection," *Information Fusion,* vol. 72, pp. 48-69, 2021.

[30] KDD99, "http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html," 1999. [Online]. [Accessed 10 02 2020].

[31] NSL-KDD, "https://www.unb.ca/cic/datasets/nsl.html," 2009. [Online]. [Accessed 10 02 2020].

[32] UNSW-NB15, "https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/," 2015. [Online]. [Accessed 10 02 2020].

[33] Y.-P. Zhou and J.-A. Fang, "Intrusion detection model based on hierarchical fuzzy inference system," in *2009 Second International Conference on Information and Computing Science*, 2009.

[34] M. Kezih and M. Taibi, "Evaluation effectiveness of intrusion detection system with reduced dimension using data mining classification tools," in *2nd International Conference on Systems and Computer Science*, 2013.

[35] P. Jongsuebsuk, N. Wattanapongsakorn and C. Charnsripinyo, "Real-time intrusion detection with fuzzy genetic algorithm," *2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology,* pp. 1-6, 2013.

[36] M. Bahrololum, E. Salahi and M. Khaleghi, "Machine learning techniques for feature reduction in intrusion detection systems: a comparison," in *2009 Fourth International Conference on Computer Sciences and Convergence Information Technology*, 2009.

[37] A. Divekar, M. Parekh, V. Savla, R. Mishra and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives," in *2018 IEEE 3rd international conference on computing, communication and security (ICCCS)*, 2018.

[38] C. Sun, K. Lv, C. Hu and H. Xie, "A double-layer detection and classification approach for network attacks," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, 2018.

[39] M. O. Miah, S. S. Khan, S. Shatabda and D. M. Farid, "Improving detection accuracy for imbalanced network intrusion classification using cluster-based under-sampling with random forests," in *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*, 2019.

[40] J. Yan, D. Jin, C. W. Lee and P. Liu, "A comparative study of off-line deep learning based network intrusion detection," in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2018.

[41] B. Subba, "A Neural Network based NIDS framework for intrusion detection in contemporary network traffic," in *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2019.

[42] Y. Lin, J. Wang, Y. Tu, L. Chen and Z. Dou, "Time-related network intrusion detection model: a deep learning method," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019.

[43] U. Sabeel, S. S. Heydari, H. Mohanka, Y. Bendhaou, K. Elgazzar and K. El-Khatib, "Evaluation of deep learning in detecting unknown network attacks," in *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*, 2019.

[44] CIC_IDS2017, "CIC-IDS2017," 2017. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html. [Accessed 20 11 2020].

[45] T. Elmasri, N. Samir, M. Mashaly and Y. Atef, "Evaluation of CICIDS2017 with qualitative comparison of Machine Learning algorithm," in *2020 IEEE Cloud Summit*, 2020.

[46] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective,* vol. 25, p. 18–31, 2016.

[47] S. Samonas and D. Coss, "The CIA strikes back: Redefining confidentiality, integrity and availability in security.," *Journal of Information System Security,* vol. 10, 2014.

[48] N. Moustafa, J. Hu and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *Journal of Network and Computer Applications,* vol. 128, p. 33–55, 2019.

[49] Y. Fu, Y. Du, Z. Cao, Q. Li and W. Xiang, "A Deep Learning Model for Network Intrusion Detection with Imbalanced Data," *Electronics,* vol. 11, p. 898, 2022.

[50] K. Jiang, W. Wang, A. Wang and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE Access,* vol. 8, p. 32464–32476, 2020.

[51] W. Xu, J. Jang-Jaccard, T. Liu, F. Sabrina and J. Kwak, "Improved Bidirectional GAN-Based Approach for Network Intrusion Detection Using One-Class Classifier," *Computers,* vol. 11, p. 85, 2022.

[52] L. Vu and Q. U. Nguyen, "Handling imbalanced data in intrusion detection systems using generative adversarial networks," *Journal on Information Technologies & Communications,* vol. 2020, p. 1–13, 2020.

[53] T. Acharya, I. Khatri, A. Annamalai and 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 & Intelligent Systems (I2CACIS)*, 2021.

[54] T. Acharya, I. Khatri, A. Annamalai and 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 & Intelligent Systems (I2CACIS)*, 2021.

[55] C. Yin, Y. Zhu, J. Fei and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access,* vol. 5, p. 21954–21961, 2017.

[56] Z. Chen, C. K. Yeo, B. S. Lee and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless telecommunications symposium (WTS)*, 2018.

[57] M. Ganesh, A. Kumar and V. Pattabiraman, "Autoencoder Based Network Anomaly Detection," in *2020 IEEE International Conference on Technology, Engineering, Management for Societal impact using Marketing, Entrepreneurship and Talent (TEMSMET)*, 2020.

[58] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei and F. Sabrina, "Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset," *IEEE Access,* vol. 9, p. 140136–140146, 2021.

[59] J. Gao, "Network Intrusion Detection Method Combining CNN and BiLSTM in Cloud Computing Environment," *Computational Intelligence and Neuroscience,* vol. 2022, 2022.

[60] A. G. Salman, Y. Heryadi, E. Abdurahman and W. Suparta, "Single layer & multi-layer long short-term memory (LSTM) model with intermediate variables for weather forecasting," *Procedia Computer Science,* vol. 135, p. 89–98, 2018.

[61] T. S. Pooja and P. Shrinivasacharya, "Evaluating neural networks using Bi-Directional LSTM for network IDS (intrusion detection systems) in cyber security," *Global Transitions Proceedings,* vol. 2, p. 448–454, 2021.

[62] Y. Imrana, Y. Xiang, L. Ali and Z. Abdul-Rauf, "A bidirectional LSTM deep learning approach for intrusion detection," *Expert Systems with Applications,* vol. 185, p. 115524, 2021.

[63] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT express,* vol. 6, p. 312–315, 2020.

[64] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, 2009.

[65] H. Liu, B. Lang, M. Liu and H. Yan, "CNN and RNN based payload classification methods for attack detection," *Knowledge-Based Systems,* vol. 163, p. 332–341, 2019.

[66] B. Cao, C. Li, Y. Song, Y. Qin and C. Chen, "Network Intrusion Detection Model Based on CNN and GRU," *Applied Sciences,* vol. 12, p. 4184, 2022.

[67] X. Ji, H. Zhang and X. Ma, "A Novel Method of Intrusion Detection Based on Federated Transfer Learning and Convolutional Neural Network," in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2022.

[68] M. Xiong, H. Ma, Z. Fang, D. Wang, Q. Wang and X. Wang, "Bi-LSTM: Finding Network Anomaly Based on Feature Grouping Clustering," in *2020 The 3rd International Conference on Machine Learning and Machine Intelligence*, 2020.

[69] S. N. Pakanzad and H. Monkaresi, "Providing a hybrid approach for detecting malicious traffic on the computer networks using convolutional neural networks," in *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, 2020.

[70] R. Yao, N. Wang, Z. Liu, P. Chen and X. Sheng, "Intrusion detection system in the advanced metering infrastructure: a cross-layer feature-fusion CNN-LSTM-based approach," *Sensors,* vol. 21, p. 626, 2021.

[71] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao and J. Chen, "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system," *Security and communication networks,* vol. 2020, 2020.

[72] L. Zhang, J. Huang, Y. Zhang and G. Zhang, "Intrusion detection model of CNN-BiLSTM algorithm based on mean control," in *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*, 2020.

[73] J. Sinha and M. Manollas, "Efficient deep CNN-BILSTM model for network intrusion detection," in *Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition*, 2020.

[74] A. Li and S. Yi, "Intelligent Intrusion Detection Method of Industrial Internet of Things Based on CNN-BiLSTM," *Security and Communication Networks,* vol. 2022, 2022.

[75] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, 2015.

[76] T. Acharya, A. Annamalai and M. F. Chouikha, "Efficacy of CNN-bidirectional LSTM hybrid model for network-based anomaly detection," in *2023 IEEE 13th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2023.

[77] L. Dhanabal and 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,* vol. 4, p. 446–452, 2015.

[78] T. Acharya, A. Annamalai and M. F. Chouikha, "Efficacy of Bidirectional LSTM Model for Network-Based Anomaly Detection," in *2023 IEEE 13th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2023.

[79] T. Acharya, A. Annamalai and M. F. Chouikha, "Optimizing the Performance of Network Anomaly Detection Using Bidirectional Long Short-Term Memory (Bi-LSTM) and Over-sampling for Imbalance Network Traffic Data," *Advances in Science, Technology and Engineering Systems Journal,* vol. 8, p. 144–154, 2023.

[80] A. Gosain and S. Sardana, "Handling class imbalance problem using oversampling techniques: A review," in *2017 international conference on advances in computing, communications and informatics (ICACCI)*, 2017.

[81] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder and N. Seliya, "A survey on addressing high-class imbalance in big data," *Journal of Big Data,* vol. 5, no. 1, December 2018.

[82] M. Al Olaimat, D. Lee, Y. Kim, J. Kim and J. Kim, "A learning-based data augmentation for network anomaly detection," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 2020.

[83] B. Bowen, A. Chennamaneni, A. Goulart and D. Lin, "BLoCNet: a hybrid, dataset-independent intrusion detection system using deep learning," *International Journal of Information Security,* vol. 22, no. 4, pp. 893-917, August 2023.

[84] B. AlOmar, Z. Trabelsi and F. Saidi, "Attention-Based Deep Learning Modelling for Intrusion Detection," in *ECCWS 2023 22nd European Conference on Cyber Warfare and Security*, 2023.

[85] X. Ma and W. Shi, "AESMOTE: Adversarial Reinforcement Learning with SMOTE for Anomaly Detection," *IEEE Transactions on Network Science and Engineering,* vol. 8, no. 2, pp. 943-956, April 2021.

[86] W. Bouzeraib, A. Ghenai and N. Zeghib, "A Multi-Objective Genetic GAN Oversampling: Application to Intelligent Transport Anomaly Detection\," 2020.

[87] A. A. Awad, A. F. Ali and T. Gaber, "An improved long short term memory network for intrusion detection," *PLoS ONE,* vol. 18, no. 8 August, August 2023.

[88] B. Padmavathi, A. Bhagyalakshmi, D. Kavitha and P. Indumathy, "An optimized Bi-LSTM with random synthetic over-sampling strategy for network intrusion detection," *Soft Computing,* vol. 28, no. 1, pp. 777-790, January 2024.

[89] R. B. Said, Z. Sabir and I. Askerzade, "CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software Defined Networking with Hybrid Feature Selection.," *IEEE Access,* 2023.

[90] F. E. Laghrissi, S. Douzi, K. Douzi and B. Hssina, "IDS-attention: an efficient algorithm for intrusion detection systems using attention mechanism," *Journal of Big Data,* vol. 8, no. 1, December 2021.

[91] Y. Liu, H. Li, W. Zhang, F. Lv and S. Si, "Intrusion Detection Based on Sampling and Improved OVA Technique on Imbalanced Data," 2023.

[92] T. Acharya, A. Annamalai and M. F. Chouikha, "Enhancing the Network Anomaly Detection using CNN-Bidirectional LSTM Hybrid Model and Sampling Strategies for Imbalanced Network Traffic Data," *Advances in Science, Technology and Engineering Systems Journal,* vol. 9, p. 67–78, 2024.

CURRICULUM VITAE

Toya Acharya

Houston, TX 77065
sathi.toya@gmail.com | 832-931-0234

Education

Ph.D. in Electrical Engineering (2024)
Prairie View A&M University, Prairie View, TX
Master of Science in Electrical Engineering (2020)
Prairie View A&M University, Prairie View, TX
Bachelor of Science in Electronics & Communication Engineering (2006)
Pokhara University, Pokhara

Certification

- **CompTIA Security + | Cisco CCNA Routing & Switching | CompTIA A+ | Cisco CCNA Cyber Ops | ISC2 CC**

Scholarship & Awards

- EC-Council **C|CT Cybersecurity** Scholarship 2023 | NSF Advanced Studies Institute **ASI in France** student 2023
- **CyberCorps** Scholarship for Service **(SFS)** 2020 | Ph.D. **Outstanding Student**- Electrical Engineering 2022
- Cisco **Global Cybersecurity** Scholarships 2017 | **Champion** Winter Classic Invitational Cluster Competition 2022

Technical Expertise

OS: Windows, XP, Linux, Kali, Ubuntu | **Scanning**: NMAP, Zenmap, Nessus, Nikto | **Enumeration**: dirb, dirbuster, enum4linux
Password Cracking: Hydra, Medusa, JTR| **Exploitation**: Metasploit | **Memory Forensic**: Volatility | **Packet Analyzer**: Wireshark,
Artificial Intelligence: Machine Learning /Deep Learning |**Data Preprocessing:** Log, data (categorical and non-categorical)
Tools: MATLAB, WEKA, Jupyter Notebook | **Information Gathering**: OSINET, Google hacking, HTTtrack, theHarvester
Programming/Scripting: Bash Scripting, Python, Pandas, Numpy, Keras, SKlearn, C, PyCaret, Assembly Language
Vulnerability Assessment: Nessus, NMAP | **Malware Analysis:** Static Malware Analysis, Dynamic Malware Analysis

Internship

Sandia National Laboratories, Livermore, CA 94551 | (06/13/2022 – 02/25/2024)
Intern Year-Round

- Labelled malware data using semi-supervised machine learning.
- Analyzed the performance of malware datasets using machine learning.
- Anomalies detection on parent-child process dataset using the DL method.
- Developed an unsupervised DL model, which updates based on the data given to it.

Naval Supply Systems Command (NAVSUP) HQ, Mechanicsburg, PA 17050 | (07/01/2021 -8/24/2021)

Summer 2021 intern

- Performed research on Commodity requirements For Military Work Force
- Performed research on cyber risk on critical infrastructure.

Experiences

SECURE Cybersecurity Center of Excellence | Prairie View A & M University, TX | (05/07/2020 - 09/30/2020) (02/26/2024 --)

Graduate Research Assistant

- Performed research on Network Intrusion Detection for the Center for Cybersecurity Research
- Analyzed Network Intrusion Detection System dataset using machine learning.

Electrical & Computer Engineering | Prairie View A & M University, Prairie View, TX | (10/28/2022 - 05/31/2023)

Graduate Assistant Teaching

- Taught undergraduate-level courses.
- Mentored students individually or in groups, including holding office hours.
- Supported instructional labs.

Pokhara Engineering College, Pokhara, Nepal | (09/07/2007 – 08/15/2011)

IT Support Technician/Instructor

- Installed and configured LAN network, hardware, and software
- Provided basic end-user troubleshooting and desktop support.
- Provided class lectures, assisted course projects and lab work, and graded students' tasks.

Leadership, Organization, and Participation

Student Manager, Team Prairie View A & M for Winter Classic Invitational Cluster Competition (2022)
Sandia's **Forensic Incident Response Exercise**, known as Tracer FIRE (2022)
DOE Cyber Fire Training on **Network Archaeology**, **Digital Forensics**, **Reverse Engineering (**2022)
Institute of Electrical and Electronics Engineers (**IEEE) student member (**2021)
InfraGard Houston Chapter Member (2020)

Publications

- **T. Acharya**, I. Khatri, A. Annamalai and M. F. Chouikha, "Efficacy of Machine Learning-Based Classifiers for Binary and Multi-Class Network Intrusion Detection," *2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), 2021*, pp. 402-407, doi: 10.1109/I2CACIS52118.2021.9495877.
- **T. Acharya**, I. Khatri, A. Annamalai and M. F. Chouikha, "Efficacy of Heterogeneous Ensemble Assisted Machine Learning Model for Binary and Multi-Class Network Intrusion Detection," *2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), 2021*, pp. 408-413, doi: 10.1109/I2CACIS52118.2021.9495864.
- **T. Acharya**, A. Annamalai and M. F. Chouikha, "Efficacy of Bidirectional LSTM Model for Network-Based Anomaly Detection," 2023 IEEE 13th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, Malaysia, 2023, pp. 336-341, doi: 10.1109/ISCAIE57739.2023.10165336.
- **T. Acharya**, A. Annamalai and M. F. Chouikha, "Efficacy of CNN-Bidirectional LSTM Hybrid Model for Network-Based Anomaly Detection," 2023 IEEE 13th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, Malaysia, 2023, pp. 348-353, doi: 10.1109/ISCAIE57739.2023.10165088.
- **T. Acharya**, A. Annamalai, M.F. Chouikha "Enhancing the Network Anomaly Detection using CNN-Bidirectional LSTM Hybrid Model and Sampling Strategies for Imbalanced Network Traffic Data," Advances in Science, Technology and Engineering Systems Journal, vol. 9, no. 1, pp. 67-78 (2024), DOI: 10.25046/aj090107
- **T. Acharya**, A. Annamalai, M.F. Chouikha "Optimizing the Performance of Network Anomaly Detection Using Bidirectional Long Short-Term Memory (Bi-LSTM) and Over-sampling for Imbalance Network Traffic Data", Advances in Science, Technology and Engineering Systems Journal, vol. 8, no. 6, pp. 144-154 (2023), DOI: 10.25046/aj080614
- **ACHARYA, T**., Akujuobi, C. M., Chouikha, M. F., & Annamalai, A. (2022, March). Simulation of Asymmetric digital subscriber line (ADSL) using the Discrete Wavelet Multitone Modulation (DWMT). In *2022 ASEE Gulf Southwest Annual Conference*.

- Khatri, I., **Acharya, T**., Annamalai, A., & Chouikha, M. (2021, August). Higher Order Statistics of channel capacity in κ-μ fading channel. In *2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN)* (pp. 35-40). IEEE.