IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 November 2023, accepted 30 November 2023, date of publication 7 December 2023,
date of current version 13 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3340142

RESEARCH ARTICLE

CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking With Hybrid Feature Selection

RACHID BEN SAID[1], ZAKARIA SABIR[2], AND IMAN ASKERZADE[1]
[1]Department of Computer Engineering, Ankara University, 06830 Ankara, Turkey
[2]ILM Department, National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra 14000, Morocco

Corresponding author: Rachid Ben Said (bensaid@ankara.edu.tr)

This work was supported in part by TÜBİTAK ULAKBİM High Performance, Grid Computing Centre.

ABSTRACT A Software-Defined Network (SDN) was designed to simplify network management by allowing the control and management of the entire network from a single place. SDN is commonly used in today's data center network infrastructure, but new forms of threats such as Distributed Denial-of-Service (DDoS), web attacks, and the U2R (User to Root) attack are significant issues that might restrict the widespread adoption of SDNs. Intruders are attractive to SDN controllers because they are valuable targets. An SDN controller can be hijacked by an attacker and used to route traffic in accordance with its own needs, resulting in catastrophic consequences for the whole network. While the unified vision of SDN and deep learning methods opens new possibilities for the security of IDS deployment, the effectiveness of the detection models is dependent on the quality of the training datasets. Even though deep learning for NIDSs has lately shown promising results for a number of issues, the majority of the studies overlooked the impact of data redundancy and an unbalanced dataset. As a consequence, this may adversely affect the resilience of the anomaly detection system, resulting in a suboptimal model performance. In this study, we created a hybrid Convolutional Neural Network (CNN) and bidirectional long short-term memory (BiLSTM) network to enhance network intrusion detection using binary and multiclass classification. The effectiveness of the proposed model was tested and assessed using the most frequently used datasets (UNSW-NB15 and NSL-KDD). In addition, we used the InSDN dataset, which is specifically dedicated to SDN. The outcomes demonstrate the efficiency of the proposed model in achieving high accuracy and requiring less training time.

INDEX TERMS Network intrusion detection system (NIDS), software-defined networking (SDN), CNN-BiLSTM, deep learning.

I. INTRODUCTION

Software-defined networking (SDN) is the result of earlier proposals that were mainly concerned with both programmable networks and control-data plane separation [1]. In the SDN paradigm, the controller, which represents a logically centralized controlling point, successfully manages and gathers flow-based statistics via a protocol such as the OpenFlow protocol maintained by the Open Networking Foundation (ONF), a nonprofit industry consortium that offers support and ensures various improvements in the SDN field [2], [3]. Even though SDN has many advantages, there may be security issues which have to be assured in order to be widely used. When a security breach occurs in a traditional network, the harm is restricted mostly to a single node or segment; however, an intrusion on an SDN controller might have far-reaching impacts on the entire network. In the situation that the controller is taken down by an attacker, the network

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

may be vulnerable to failures. In addition, an attacker may inundate the network with hazardous cyber-attacks, such as Distributed Denial of Service (DDoS) and Denial of Service (DoS) attacks. As a result, genuine queries may be refused due to the heavy consumption of channel bandwidth and network resources [4].

The dynamic and configurable nature of SDN may increase the false detection of attacks. An SDN controller may be hijacked by an attacker and used to route traffic according to its own needs, resulting in consequences for the whole network. Although IDS is a legitimate way to make sure a network is safe and can detect and stop intrusions, it is not directly clear whether SDN's more complicated flow management may increase the chances of service failures. Although the purpose of an IDS is to monitor network traffic and warn the administrator of any malicious threats in the network, the particularities of SDN may raise exposure to sophisticated cyber-attacks, which has led to concerns about developing innovative IDS approaches to secure SDN, protect user communication [5], and reveal new security problems that appear.

Feature selection is becoming an important phase of the preparation for network classification problems. The selection of features helps us reduce and remove irrelevant and redundant features from the main database that have no impact on the classification results. The method of enabling feature selection selects a subset of the original dataset with some criteria that contain the properties of the original dataset. According to Kantardzic [6], classification improves when the features of large datasets are reduced by using basic techniques. There are two types of feature selection algorithms: filter-based and wrapper-based. Filter methods are computationally efficient, have fast processing speeds, and are less likely to be overadjusted [7]. Common methods for filter selection include ANOVA, Chi-square, and Pearson's correlation. On the other hand, wrapper methods provide the best subset of relevant functions, but they require more computational time, which decreases system performance. Recursive selection, forward selection, and backward elimination are commonly employed by wrappers. As network traffic increases, intrusion detection systems face problems in terms of data dimensions and complexity.

One of the proposed solutions for improving NIDS (network intrusion detection system (NIDS) monitoring is to use Deep Learning (DL) models. While the unified vision of SDN and deep learning methods opens new possibilities for the security of IDS deployment, the effectiveness of the detection method is dependent on the quality of the training datasets. Even though deep learning for NIDSs has lately shown promising results for a number of issues, the majority of the studies overlooked the impact of data redundancy and an unbalanced dataset. As a consequence, this may adversely affect the resilience of the anomaly detection system, resulting in a suboptimal model performance.

Our model aims to enhance SDN security by employing a convolutional neural network–bidirectional long short-term memory (CNN–BiLSTM) hybrid algorithm. The previous DL models required a high number of training parameters to be effective. Forcing the model to use a large number of parameters may slow down its training phase and increase its computational cost. As a result, it adds extra computing overhead to the SDN architecture. Our approach suggests using deep learning over SDN to optimize NIDS deployment. The NIDS implementation of the SDN controller features a deep learning approach for network monitoring. To better identify anomalies, this study proposes the use of tree-based deep-learning approaches. Ten features were used in a multi-class classification to determine whether or not an intrusion has occurred and to classify the kind of intrusion. In comparison with other studies, previous works have used deep learning algorithms in NIDS to secure SDN, such as [8] and [9]. However, they only used general datasets, such as NSL-KDD and UNSW-NB15, which are not dedicated to SDN. Other authors [10] used the InSDN dataset, but their model employed CNN-LSTM, which is considered weak compared to the CNN-BiLSTM model. This model used BiLSTM, which has different sequences. As each sequence of BiLSTM moves forward and backward, it has two LSTM layers. This compensates for the absence of contextual semantic information in LSTM. Each single time in the input sequence of the output layer is covered by a bidirectional structure, which provides information about both the past and future. The BiLSTM Networks employed in our model were used to better find LSTM dependent features and improve the accuracy of classification [11].

The major contributions of this paper are flowing:

- We propose an approach that includes a hybrid model combining a convolutional neural network (CNN) with Bidirectional Long Short-Term Memory (BiLSTM). We used the L2 regularizer and dropout (0.5) techniques in the training model. Our hybrid CNN-BiLSTM model achieved high classification performance.
- Numerous experiments were conducted to validate the performance of the proposed hybrid model for multi-class and binary classifications. Additionally, we balanced the datasets using random oversampling, after that we used a random forest classifier and recursive feature elimination algorithm to select the highly important features so that the proposed model could train those features with high performance. Each experiment was validated by using the InSDN dataset.
- The efficiency of the proposed model was verified using two distinct benchmark datasets: NSL-KDD and UNSW-NB15.
- The efficacy and performance of suggested deep learning algorithms were tested using a set of measures including accuracy, recall, F1-score, precision, and training time. The findings indicate that our model outperforms the CNN, AlexNet, LeNet5, and CNN-LSTM models for the majority of the evaluation measures.

The remainder of this paper is organized into four sections. Section II outlines several related works concerning network

intrusion detection. The proposed method is presented in detail in Section III. Section IV is devoted to the presentation of the experimental results as well as the analysis achieved on the dataset. Section V summarizes the results of this study and discusses future work.

II. RELATED WORK

Deep Learning is a subfield of machine learning that has been used successfully in a variety of research areas such as image processing, face recognition and natural language processing. Recently, various deep learning methods have been evaluated in NIDS research; specifically, CNN and LSTM models have been experimented with because of their potential ability to represent contextual and sequence or temporal information, respectively, in the training datasets.

Elsayed et al. developed a hybrid intrusion detection system that combines CNNs and LSTMs. The proposed model can capture both the spatial and temporal features of the network traffic. In this study, the authors used two regularization techniques, L2 regularization and dropout, to deal with the overfitting problem [8]. The authors used a recently published novel dataset of NIDS in SDN (InSDN) [12]. The proposed model improves the intrusion detection performance for zero-day attacks, and the results achieved by this model can reach 96.32% accuracy.

Kaiyuan et al. proposed a network intrusion detection algorithm that combines hybrid sampling with deep hierarchical networks, that is, CNN and BiLSTM. In the first stage, they used One-Side Selection (OSS) to reduce the noise samples in the majority category and then increased the minority samples using the synthetic minority oversampling technique (SMOTE) [9]. Using this hybrid technique, they balanced the dataset so that the model could learn features of minority samples and greatly reduce the model training time. In the second stage, they used a CNN to extract spatial features and a BiLSTM to extract temporal features. By using this method, the proposed model achieved good classification accuracy in two datasets, NSL-KDD and UNSW-NB15, with 83.58% and 77.16%, respectively. However, this study did not deal with NIDS in SDN.

On the other hand, the incorporation of data preprocessing methods with machine learning, such as data reduction, data augmentation, and feature selection, into SDN has received interest. In [10], a method was suggested to address the difficulties in the KDD Cup 99 dataset by conducting large-scale experimental research with the NSL-KDD dataset to achieve high accuracy in intrusion detection. The experiment was carried out on five effective machine learning algorithms (Naïve Bayes, CART, RF, SVM, and J48). The correlation feature selection approach was applied to minimize the feature complexity, providing only 13 features in the NSL-KDD dataset.

In TSDL [13], a deep neural network model with two stages was constructed and suggested for NIDS, employing a piled auto-encoder combined with SoftMax in the output layer as a classifier. TSDL was created and developed to identify attacks using multiclass classification. Down-sampling and several pre-processing techniques have been used on various datasets to increase the detection rate and monitoring effectiveness. The UNSW-NB15 detection accuracy was 89.134 %. In this study, UNSW-NB15 was the only dataset used to verify the effectiveness of the proposed model and did not treat the network intrusion detection systems in the SDN case.

Authors of [14] suggested many neural network models for NIDS, including seq2seq structures using LSTM (Long-Short-Term-Memory), variational auto-encoder, and fully connected networks. Several datasets, including NSL-KDD, KYOTO-HONEYPOT, MAWILAB, and UNSWNB15, were used to create and execute the suggested technique to distinguish between malicious and legitimate packets in the network. Different preprocessing methods have been utilized, such as normalization and one-hot encoding, to prepare data, smooth training, feature manipulation, and selection in neural networks. These parameters are intended but are not limited to allowing neural networks to learn complicated characteristics from the diverse scope of a packet.

A deep neural network architecture [15] was built on the KDD cup99 to check intrusion attempts using four hidden layers. Data preparation and decreased data usage were achieved using feature scaling and encoding. For several datasets, more than 50 characteristics were utilized to complete this assignment. Therefore, advanced hardware GPUs were employed to handle such a large number of characteristics while reducing the training time. Although the authors obtained a high accuracy, the NIDS in SDN was not the focus of their study.

Tang et al. [16] used a Deep Neural Network (DNN) to detect flow-based anomalies in SDN. To reduce the computational cost of attack detection, only six main characteristics from the NSL-KDD database were used. The structure of the DNN model consists of three hidden layers, each with 12, 6, and 3 neurons. The suggested model has a global accuracy of 75%, which is less than the threshold required for real-world environment implementations. Using the same NSL-KDD database, the authors improved their suggested model using a Gated Recurrent Unit (GRU) to achieve better accuracy [17]. The upgraded model detection rate increased to 89%. Although the authors improved their model, they did not use feature selection, which may have significantly enhanced the obtained results. Table 1 summarizes and compares the most important studies on this topic.

The earlier DL models, on the other hand, needed an enormous number of training parameters (as all neighboring layers are entirely linked together). Using a significant number of parameters may hold back the training phase and increase the computing charge of the detection model. As a result, in an SDN architecture, it adds unnecessary computational load. By employing deep learning over SDN, the suggested approach improves NIDS implementation. It incorporates a deep learning technique for network monitoring into the SDN controller's NIDS implementation. In this

TABLE 1. Different deep learning and machine learning techniques for NIDS in traditional networks and SDNs.

Ref.	Dataset	Method	Number of selected features	Type of attacks	The highest accuracy achieved		Type of network
					Binary class classification	Multi-class classification	
[18]	NSL-KDD, UNSW-NB15 and CIC-IDS2017	CNN and GRU	-	Anomaly detection	93.10% on NSL-KDD, 91.21% on UNSW-NB15 90.17% on CIC-IDS2017	-	Traditional
[19]	Owned	ANN, LSTM, and CNN	3-flow-based features	Crossfire	87% for LSTM 80% for CNN	-	SDN
[16]	Owned:800000+Packets	Neuro Evolution of Augmenting Topologies (NEAT)	3-packet-level features	Worm, DoS	-	90%	Traditional
[20]	NSL-KDD	Meta-Heuristic Bayesian Network Classification (MHBNC)	Extraction of features and pre-processing	DoS, U2R, Probe and R2L	Not reported	Not reported	Traditional
[21]	KDD-Cup 1999	Restricted Boltzmann Machine	41 features	General anomaly	Precision rate 94%	-	SDN
[22]	KDD99	Decision Tree	10 feature and 15 features	Anomaly or benign	82.48% for 10 features and 91.17% for 15 features	-	Traditional
[23]	NSL-KDD	SMO, J48 and NB	41 features	DDoS, U2R, R2L, Probe	-	97%,98.76%,95.11%	SDN
[16]	NSL-KDD	Deep neural network (DNN)	6 features	DoS, R2L, U2R, and Probe	-	75.75%	SDN
[24]	Real Traffic	Maximum entropy-TRW-Rate-limiting NETAD	-	DoS and Port scan	85%	-	SDN
[25]	NSL-KDD	Binary Bat algorithm (BBA) and Entropy	DDoS 27 features, Probe 33 features, R2L 32 features, U2R 26 features.	DDoS, Probe, R2L, U2R	-	91.10%	SDN
[26]	DARPA2000	SVM	-	DDoS	-	95.11%	SDN
[27]	Real Traffic	Auto-encoder	-	DDoS	99%	95.96%	SDN
[28]	Real Traffic	Entropy	-	DDoS	96%	-	SDN
[29]	DARPA 1999	C4.5Decision tree	-	DoS and Probe	-	96%	SDN
[30]	Real Traffic	DT, BN, NB,C4.5	-	Brute force	91.68%	-	SDN
[31]	NSL-KDD	RNN	-	Anomaly-based IDS	Not reported	Not reported	SDN
[32]	CAIDA	Rule-based	-	DDoS	-	-	SDN
[33]	NSL-KDD, UNSW-NB15 and KDDCup 1999	RNN	-	Anomaly and benign	97.39% on NSL-KDD	-	SDN
This study	UNSW-NB, NSL-KDD and InSDN	CNN-BiLSTM	10 features	BFA, U2R, DDoS, DoS, R2L, Botnet, Web Attack, Probe, and Normal	97.77 % on InSDN, 95.96% on NSL-KDD, 93.51 % on UNSW-NB15	97.12 % on InSDN, 98.42% on NSL-KDD, 84.23 % on UNSW-NB15	SDN

research, improved deep learning methods for anomaly detection were suggested. We employed ten features to perform a multi-class classification that involves identifying whether or not there is an intrusion and categorizing the kind of intrusion.

III. METHODS AND TOOLS USED IN EXPERIMENT

Each component in the NIDS structure and its purpose are discussed and explained in this section. The NIDS is designed as the SDN controller. The purpose of this study is to show the usage of the SDN architecture as a network infrastructure for a NIDS. Figure 1 depicts the architecture of NIDS in SDN, which consists of three major components: Anomaly Mitigator, Anomaly Detector and Flow Collector.

The Anomaly Mitigator can make flow choices based on the results of the Anomaly Detector (e.g., forward or drop the flow). The model was used as the core of the anomaly detector. This module runs a trained model, gets network information, and determines whether or not a flow is an anomaly. A message or timer function triggers the flow collector, which aggregates all flow statistics, including the source and destination ports, and the source and destination IP protocol. The Anomaly Detector module will get all aggregated characteristics.

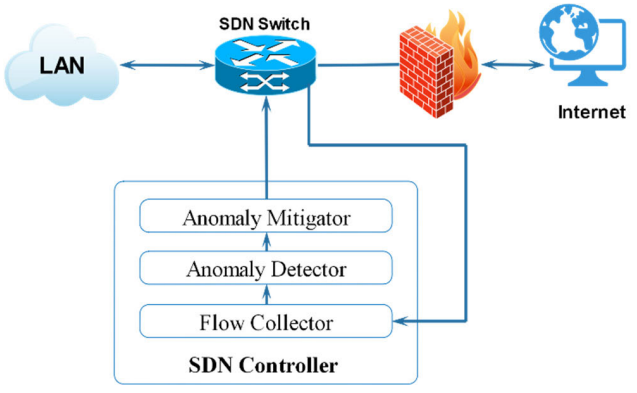

FIGURE 1. NIDS architecture in the SDN controller.

The SDN model, as can be seen in Figure 2, the architecture of NIDS in SDN, which consists of three major components: The application layer's primary responsibility is to handle all network management activities that may be carried out with the help of a network intrusion detection system and an SDN controller. The control layer is in charge of controlling activities and managing traffic data by initiating or terminating every network flow. The infrastructure layer comprises two major components: hardware and software, such as routers and OpenFlow switches.

Figure 3 depicts the overall framework of the network intrusion detection model in SDN, which includes these approaches. First, we start with transformation and cleaning that content normalization and numerical process. In addition, we balanced the dataset using random oversampling. Subsequently, we performed feature selection using a random forest classifier algorithm to obtain the highly important

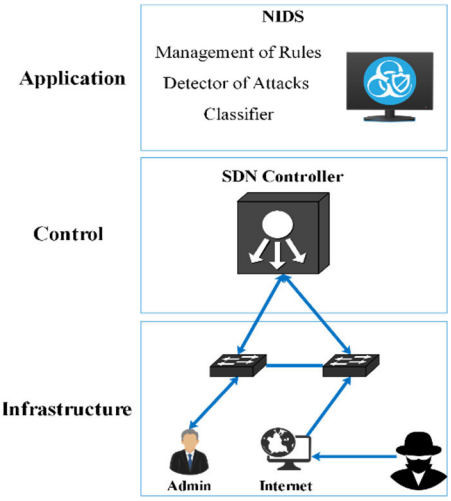

FIGURE 2. Architecture of NIDS in SDN.

features. Finally, we deployed the hybrid model in the training and testing datasets to obtain multiclass classification results.

A. DATASET DESCRIPTION

The InSDN [12] dataset is widely regarded as one of the first methods for generating a complete dataset for evaluating an IDSs for SDNs. We used the attack collection dataset to verify the performance of our model. The collection contains attacks with properties that are unique to SDN networks. Choosing an SDN-specific dataset is critical for conducting a thorough evaluation of SDN-based threat-detection techniques. Utilizing non-specific SDN datasets might create a compatibility issue because the attack vector implementation should consider the new architecture of the network [34]. Furthermore, certain attacks operate differently on SDN networks than on traditional networks. For instance, an attacker may use techniques like "IPsweep" and "Portscan" to flood the SDN controller's bandwidth with unknown packets, resulting in a DDoS attack. Although the abovementioned attacks do not qualify as DDoS by traditional definitions, they may be utilized to produce a large number of data packets in an SDN network. Numerous records were included in the InSDN collection.

SDN-specific attacks are included in the InSDN dataset as threats common to conventional networks, and SDN-specific attacks on various forms of standard traffic. DoS, DDoS, Probe, Botnet, Exploitation, Brute force, and web attacks are among the several forms of attacks. In addition, various internal and external attack vectors were used to mimic real-world attack scenarios. We split each of the three datasets into 70% for training and 30% for testing.

Furthermore, we considered two other publicly accessible intrusion detection datasets that have been frequently used in various studies: NSL-KDD and UNSW-NB15. To see if our proposed model produces generalizable results, we trained and tested it on the NSL-KDD dataset, followed by the UNSW-NB15 dataset.

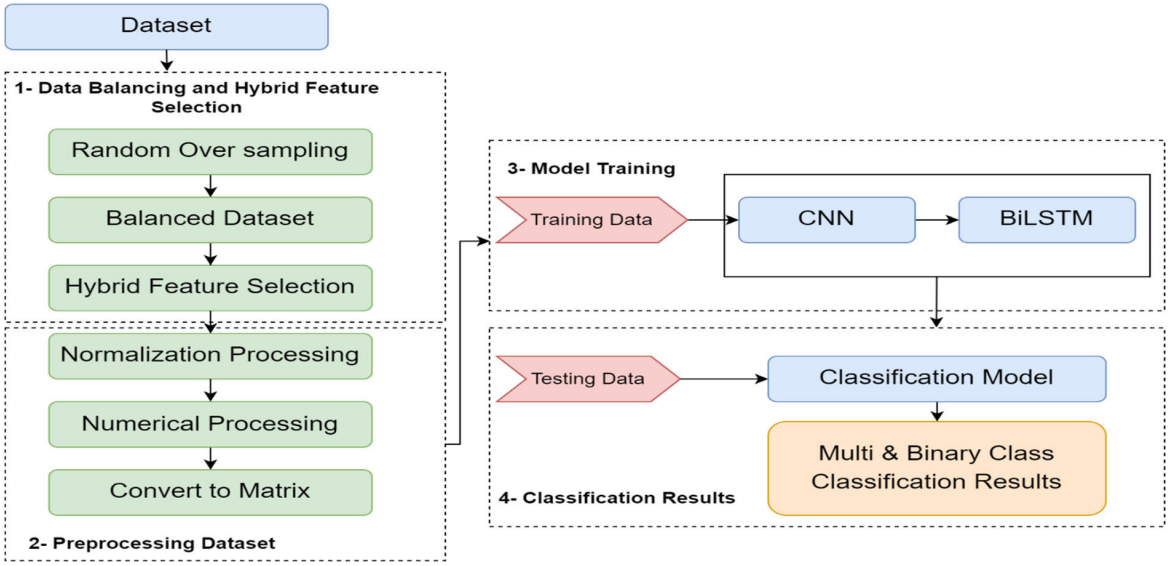

FIGURE 3. Model of the overall framework of the proposed network intrusion detection method used in SDN.

The KDD CUP99 and the NSL-KDD are well-known datasets in the context of network intrusion detection, and Revathi's research demonstrates that the NSL-KDD datasets are ideal for evaluating different intrusion detection algorithms [13]. Each incursion record in this dataset includes a 42-dimensional feature that is subdivided into a traffic-type label, 3-dimensional symbol feature, and 38-dimensional digital feature. The label mostly comprises normal data as well as the data of four categories of attack (R2L, DoS, U2R, and Probe). In this study's experiments, the test set (KDDTest) and training set (KDDTrain) from the NSL-KDD dataset were utilized as the model's test and training sets, respectively.

The Australian Centre for Cyber Security research team has produced a dataset containing the most recent updates named the UNSW-NB15 dataset [35] to address the concerns detected in the NSL-KDD and KDDCup 99 datasets. A full connection record partition comprised 82337 test connection records and 175343 train connection records linked by 10 attacks. The partitioned dataset included 42 characteristics with parallel-class labels that were typical and nine distinct intrusions.

B. DATA PREPROCESSING

The process of data processing, which is referred to as data engineering, is crucial for the success of the learning process. It involves various procedures such as column and row cleaning, feature encoding, and data normalization, which are used to preprocess the data to ensure that they are adequately prepared for analysis. All these procedures were applied to all datasets. This subsection discusses the details of these procedures as follows.

1) DROP ROWS AND MISSING VALUES

To ensure data integrity, all rows were inspected thoroughly to detect missing values. This process is a standard practice in data preparation, as the "id" column typically contains no significant data. Eliminating the "id" column allows for a focus on examining only essential features.

2) CATGEORICAL LABELENCODER ENCONDING

The process of categorical transformation holds significance in enhancing the learning capacity of classifiers that are designed to handle only numerical data. Specifically, in the context of the InSDN dataset, attributes like "proto," "service," and "state" encompass categorical information that has been converted into numeric representations.

We have opted to employ the LabelEncoder technique for encoding purposes [36]. This choice is well suited for the InSDN dataset because it assigns a distinct numerical label to each unique categorical value. This transformation enables the classifier to comprehend and derive insights from the encoded labels. While LabelEncoder differs from OneHotEncoder in that it does not generate distinct binary columns, it nonetheless preserves the categorical characteristics of the feature [37].

3) BALANCING THE DATASET

Among the several oversampling methods available, random oversampling is the simplest method for balancing an unbalanced dataset. It provides data parity by duplicating the minority class samples. This does not result in any information loss; however, the dataset is susceptible to overfitting because identical information is duplicated. In this study, we used random oversampling to deal with imbalanced datasets.

4) HYBRID FEATURE SELECTION

Data were gathered from network packets to identify intrusions. As a result, manually classifying the huge quantity of network data gathered by the system is a time-consuming operation. Apart from gathering network data, evaluating it

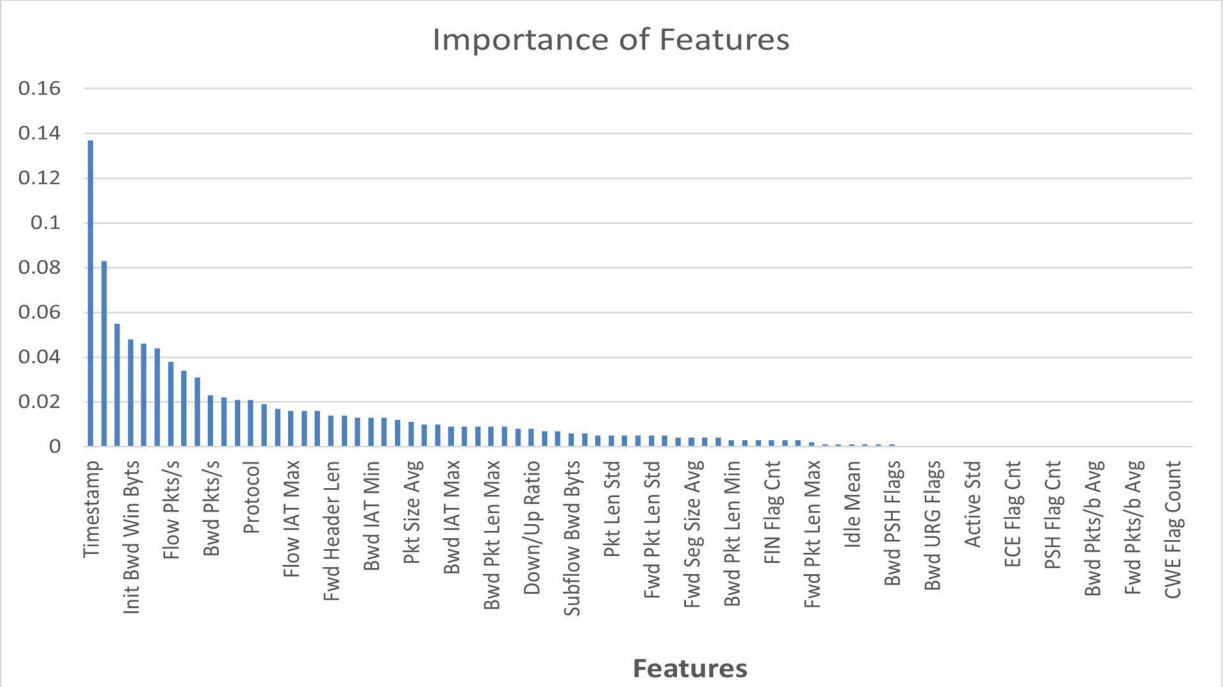

FIGURE 4. High importance of the features.

is a complicated process due to the number of behavioral patterns and attributes included in the data. To protect the network from attacks, real-time intrusion detection is necessary, which may be accomplished by mining accessible datasets for key characteristics. The reduced collection of features can significantly improve the intrusion detection rate [38]. The selection of features may be accomplished in a variety of ways. For example, filtering data that are unnecessary to the detection process in order to categorize the attacks, clustering data based on their similarity in order to uncover hidden patterns in the data for classification and deleting irrelevant features from the feature set using feature selection techniques.

We used a random forest classifier algorithm to select the high important features because it is a very accurate algorithm that performs well on massive datasets [39]. Random Forests (RF) is a classifier composed of several decision trees. It is a mixture of classification trees so that each tree is reliant on the values of a random vector selected randomly for all trees in the forest and having the same distribution [40]. When new records are received as input, the RF stores them in forest trees. Each tree provides a categorization, and the forest selects the class in which the majority of trees fall [40]. From the results of the RF classifier, we selected 10 features using Recursive Feature Elimination (RFE) because of the high accuracy achieved after different attempts while training our proposed model using different sets of selected features. Figure 4 shows the importance of each feature and Table 2 shows the feature name and the descriptions of the 10 selected features.

5) NORMALIZATION PROCESSING

The calculated value scope of continuous feature data in the InSDN dataset is noticeably different; in the dataset, the value scope of logged-in is [0,39], whereas the value scope of num compromised is [0,27]. As a result, the maximum value scope varies greatly. To make arithmetic calculations and dimension removal easier, the normalization processing approach is used to consistently and straightly map the value scope of each feature within the [0,1] interval. The normalization equation's formula (1) is as follows:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

where X_{norm} represents feature normalzed, X_{max} is the maximal feature's value and X_{min} is the minimal value.

6) NUMERICAL PROCESSING

Because the model input is a digital matrix, a hot encoding approach was utilized. The utility of this approach is to align the data in the dataset with the symbol features to a digital feature vector.

7) GENERATING A MATRIX USING NORMALIZED DATA

Each network of usable data was recorded and dimensionally converted to correspond to a grayscale image format. For example, RGB = 1 is the default; to enter the convolutional neural network (CNN), the network data are reformed into a matrix; for example, 100 feature vectors are reformed into a 10×10 matrix.

TABLE 2. Selected 10 features with their description.

Feature name	Description
'Timestamp',	Records the date and time when a particular event or data entry occurred.
'Init Bwd Win Byts',	The total number of bytes sent in the initial window in the backward direction.
'Flow Pkts/s',	Number of flow Packets per second.
'Bwd Pkts/s,	Number of backward packets per second.
'Dst Port',	Destination port number
'Protocol',	The protocol type
'Src Port',	Source port number
'Dst IP',	Destination IP address
'Fwd IAT Min',	Min and standard deviation of the time between two packets sent in the forward direction.
'Bwd Header Len',	Total bytes used for headers in the backward direction.

IV. THE PROPOSED NETWORK INTRUSION DETECTION MODEL

DL is an instance of ML. Using neural networks with multiple hidden layers, the input and output layers of the models are constructed. Each input layer node receives metadata as input and processes the data through the hidden layers of the nonlinear transformation of the input data using training adjusted weights to compute the output.

One of the major purposes of Deep Learning is that it does not require the feature extraction phase. Precisely, instead of manually selecting and extracting features, Deep Learning learns the deep structure from the original data and extracts the features automatically.

However, the redundancy of noisy features and the huge amount of data in addition to the poor classification algorithm of deep learning cause the degradation of the accuracy of detection in the NIDS (Network Intrusion Detection System) [39]. Therefore, in our proposed model, we used the RF classifier algorithm and RFE to select the optimal feature subset, as well as deep learning to ensemble the output of the multi-classifier. As a result, the accuracy of intrusion detection was improved effectively.

A. CONVOLUTIONAL NEURAL NETWORKS (CNN)

A CNN is a supervised learning approach and is one of the fundamental models often used in computer vision. The CNN can accelerate the training process because it resolves the parameter explosion problem by developing a weight-sharing principle. It's been utilized in a variety of scenarios, including image processing and face reorganization. The CNN architecture is made up of three essential parts: a pooling layer, a convolution layer, and a'' fully connected'' layer. The linear operation is applied via shifting the filter (kernel) of a given size through the outcome of the preceding layer; this is done in the convolution layer. The rectified linear unit (ReLU) activation function is the most widely used activation function in convolutional neural networks (CNNs) to introduce nonlinearity and set all negative values in the feature map to zero. A convolutional neural network (CNN) may consist of multiple convolution layers, with the initial layer specifically designed to extract basic features such as corners and edges. The ensuing levels are used for complicated feature extractions, and the pooling layer is used to minimize the feature size, thus lowering the computing costs. The last completely linked layer is utilized for classification goals.

B. BIDIRECTIONAL LONG-SHORT TERM MEMORY (BiLSTM)

A Recurrent Neural Network (RNN) is a type of neural network designed for sequential data processing. However, it exhibits instability due to challenges related to gradient vanishing and exploding. HochReiter et al. used Long Short-term memory based on RNN [41]. The difficulties of gradient disappearance and gradient explosion in the RNN training process were successfully solved by adding the gate mechanism and memory unit.

In contrast to LSTM, bidirectional long-short term memory allows information to move on both sides; forward to backward and backward to forward employing two hidden states, this can aid BiLSTM to learn the context. With the help of these two-way directions, incoming data from both the future and the past will be kept. In the backward, the LSTM state is used to acquire previous information. This architecture aids the network in retaining previous and subsequent information. In Bi-LSTM, the first layer's sequence output is the second layer's input, while the second layer's sequence output is the concatenation of the final unit output of the forward and backward layers. The final result of the piled BiLSTM layers is given by h:

$$h = [h_{forward}, h_{backward}] \tag{2}$$

C. THE PROPOSED MODEL

1) EXPERIMENT SETUP

In the experiment, we used the Truba platform from TUBITAK ULAKBIM. As the backend, TensorFlow is under the Jupyter Notebook, encoded with Python and Keras. The experimental setup is illustrated in Table 3.

2) BUILDING THE HYBRID MODEL

This section describes the architecture of the CNN-BiLSTM hybrid model for learning and classifying network data in time and space. Figure 5 depicts the overall network architecture, which was divided into three phases. The first phase

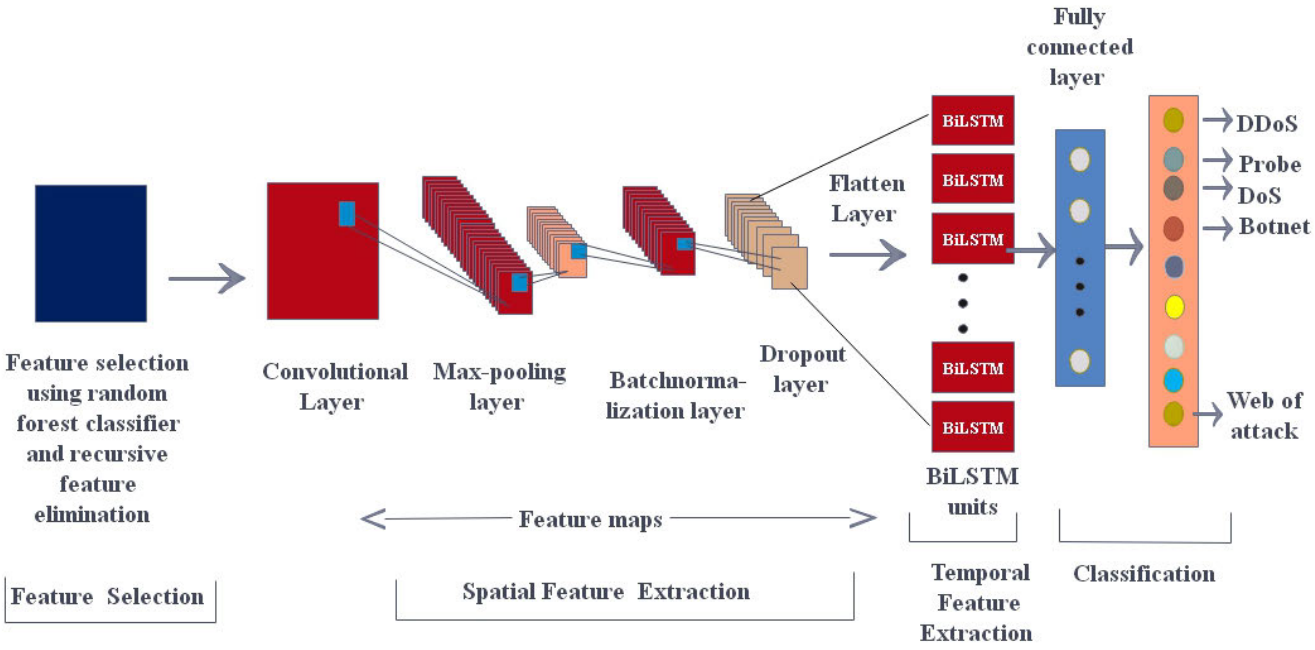

FIGURE 5. Architecture of the CNN-BiLSTM model.

TABLE 3. Experimental setup for model training.

Project	Environment
GPU	12 core x 2 CPU + 2x Nvidia M2090
Memory	256 GB + 6 GB GDDR5
CPU	Xeon E5-2680 v3 2.50 GHz
Description	Levrek-CUDA Cluster
Framework	Keras2.2
Architecture:	*x86_64*
CPU op-mode(s):	*32-bit, 64-bit*
CPU(s):	*48*
Vendor ID:	*GenuineIntel*
CPU family:	*6*
Model name:	*Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz*

consists of feature selection, as explained above, while the second and third phases are responsible for extracting and deriving data, respectively. Both CNN and BiLSTM are well-known deep learning algorithms. CNNs are capable of extracting spatial dimensional information from data. BiLSTM is unique in that it retains contextual background data for an extended period of time and enables the derivation of data characteristics at the temporal level. It is vital to examine the feature link at the spatial level while extracting features from a network intrusion detection system. As a result, this model integrates CNN and BiLSTM to extract features and then builds a hybrid model.

Due to the fact that the CNN and BiLSTM mechanism inputs have different formats, the derived spatial features are changed at the CNN output to conform to the BiLSTM network's input format. In our case, the completely linked layer of the CNN produced 1×128, 1×64, and 1×32 feature vectors as the output. When applied to the input layer of the BiLSTM model, the input value is fixed at 70. One layer of BiLSTM units was used to extract temporal features. Three convolution layers were created using kernels sizes of 5,3 and 5. Each convolution layer is a Max Pooling layer with a size of 2×2 to minimize the dimensionality of the batch normalization and map features. The high-dimensional characteristics extracted in the CNN stage are passed to the next stage, which has three layers: a fully connected layer, a BiLSTM layer, and an output layer. In both the BiLSTM and fully connected layers, there were 10 neurons for multiclass classification and one neuron for binary-class classification. Finally, at the output, the sigmoid layer was employed to represent the likelihood of each input flow for classification purposes.

We employed the L2 Regularizer and dropout approaches to mitigate the impact of overfitting and improve the detection model's capacity in unseen data. We ran multiple tests to find the appropriate regularization hyperparameter λ given the dropout technique's probability values. The value of λ is determined to 0.1 for the L2 Regularizer, and a dropout with a probability P of 0.5 is employed after the following pooling layer and the entirely connected layer, respectively.

TABLE 4. Hyperparameters settings.

Parameters	Values
Kernels size	5.3.5
Filters	128.64.32
Size of the pool	2
Size of BiLSTM output	122
Regularization of the kernel	L2(0.1)
Weight restrictions	2
Optimizer	Adam
Activation function	Relu
Batch normalization	yes
Dropout	0.5
Loss function	Categorical_crossentropy for multi class classification Binary_crossentry for binary class classification
Learning rate	0.001
Batch-size	128

We begin with a low dropout probability and progressively increase it because a dropout might induce some errors inside the learning model, and we want to limit the propagation of this loss to the subsequent layers. We trained all the models using k-fold cross-validation with 2-fold and 20 epochs for each fold.

3) HYPER-PARAMETER SETTING

In some circumstances, the model may result in lower accuracy or even overfitting or underfitting. Performing hyperparameter adjustment is crucial for achieving high model performance. For that reason, the randomized search approach was utilized to refine the hyper-parameters and improve accuracy. Table 4 shows the hyperparameter values for the proposed model.

4) EVALUATION METRIC

The model was tested using a standard performance assessment. The model's accuracy, precision, recall, and f1-score are the four principal indicators to evaluate. These four indicators are basically derived from the four basic attribute of the confusion matrix as follows:

a. True Positive (TP): indicates that attack data is correctly classified as an attack.
b. True Negative (TN): indicate that normal data is correctly classified as normal.
c. False Positive (FP): indicate that normal data is incorrectly classified as an attack.
d. False Negative (FN): indicate that attack data is incorrectly classified as normal.

The evaluation indicators accuracy (3), Precision (4), Recall (4) and F1-score (5) below are the principal indicators to evaluate our model:

Accuracy(A):

$$A = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

Precision (p):

$$p = \frac{TP}{TP + FP} \tag{4}$$

Recall (r):

$$r = \frac{TP}{TP + FN} \tag{5}$$

F1-Score (f1):

$$f1 = \frac{2\,(TP + FP)\,(TP + FN)}{TP} \tag{6}$$

V. EXPERIMENT RESULTS AND DISCUSSION

A. BINARY CLASS CLASSIFICATION RESULTS FOR ALL DATASETS

Several deep learning and machine learning algorithms are currently being used to detect network intrusions. In network intrusion detection, random forests and standards convolutional neural networks are widely used. As a result, the algorithms in this paper are compared to the classic classification algorithms widely used in intrusion detection. In this study, classification performance was compared using AlexNet, LeNet5, CNN, and CNN-LSTM networks.

The suggested model was compared to numerous deep learning models, including CNN, ALexNet, LeNet5, and CNN-LSTM, according to the study with the optimal hyperparameter tuning. Additionally, it was tested in the InSDN, UNSW-NB15, and NSL-KDD datasets. Table 5 and Figure 6 show the binary classification for each model in InSDN dataset. The CNN-BiLSTM is determined to have the top performance for binary classification across all evaluation metrics. Our model has an average accuracy of 97.77 %. The suggested model outperformed the previous models i.e., CNN, AlexNet, LeNet5, and CNN-LSTM by 2.11 %, 31.19 %, 5.68 %, and 2.74 %. This illustrates the significance and ability of CNN-BiLSTM for efficiently detecting anomalies with the least number features.

We also conducted tests on the UNSW-NB15 and NSL-KDD datasets to further validate the suggested technique in this research. Table 5 summarizes the experimental findings. The classification performance of all classifiers is between 84 and 95 %. In classification performance, the approach described in this research outperformed AlexNet, LeNet5, CNN, and CNN-LSTM by 8.13 %, 9.23 %, 9.04 %, and 7.62 %, respectively, on the UNSW-NB15 dataset.

TABLE 5. Binary classification results for all dataset and the running time.

Models	Dataset	metrics				
		Accuracy (%)	F1-score (%)	Precision (%)	Recall (%)	Training time (s)
CNN	InSDN	95.66	91.27	94.89	89.16	421.15
AlexNet		66.58	64.52	75.83	74.83	1633.62
LeNet5		92.09	90.07	93.34	89.01	**242.01**
CNN-LSTM		95.03	94.6	93.59	96.11	736.56
CNN-BILSTM		**97.77**	**97.51**	**99.85**	**95.28**	**978.42**
CNN	NSL-KDD	84.1	80.35	85.36	79.6	369.9
AlexNet		86.1	75.97	74.17	77.86	456.87
LeNet5		86.02	81.9	85.8	81.11	1264.9
CNN-LSTM		92.25	93.49	89.3	98.04	1023.65
CNN-BILSTM		**95.96**	97.21	99.12	94.6	1236
CNN	UNSW-NB15	84.47	73.38	71.6	75.61	297.3
AlexNet		85.38	75.34	72.37	81.32	627
LeNet5		84.28	70.45	69.08	72.33	1411
CNN-LSTM		85.89	71.3	69.16	75.01	1229
CNN-BILSTM		**93.51**	76.04	72.3	80.63	1885

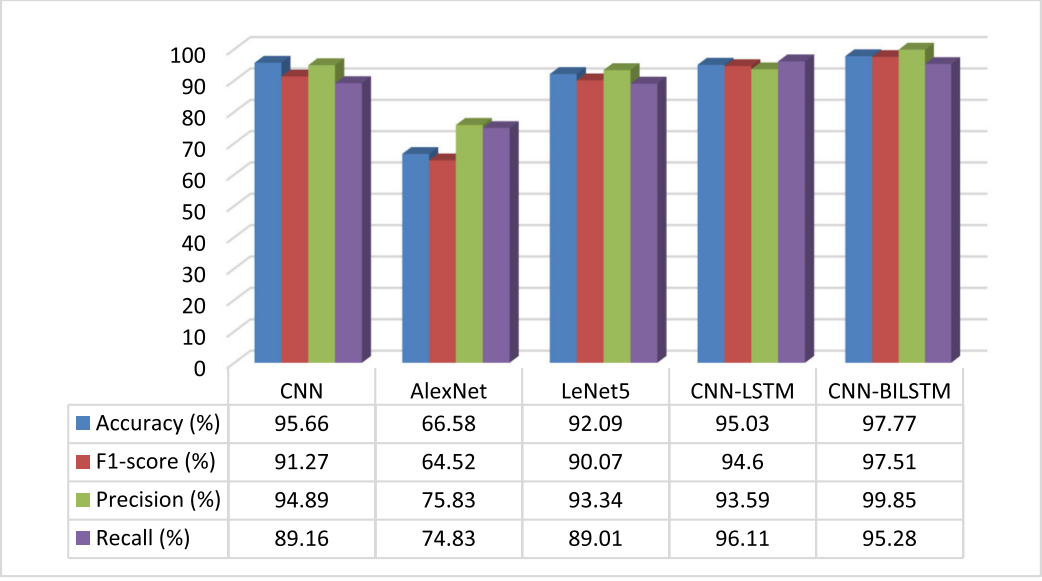

FIGURE 6. Binary class classification results comparison on InSDN dataset.

It also outperformed the same models in NSL-KDD dataset by 9.86 %, 9.94 %, 11.86%, and 3.71% respectively.

As illustrated in Table 5, among all classification models in the InSDN dataset, the CNN-BiLSTM produced the best detection accuracy. CNN, AlexNet, LeNet-5, CNN-LSTM, and CNN-BiLSTM take 421.15s, 1633.62s, 242.01s, 736.56s, and 978.42s of training time, respectively. Although CNN-BiLSTM took more training time than LetNet-5, this was tolerable in consideration of the much higher accuracy. CNN-BiLSTM performed much better than AlexNet.

B. MULTI-CLASS CLASSIFICATION RESULTS

1) RESULTS AND DISCUSSION ON INSDN DATASET

Table 6 provides detailed findings for multi-class classification using several classifiers on the InSDN dataset. In terms of different metrics, the performance of various attacks varies significantly amongst classifiers.

- The accuracy of different models was depicted by figure7, and the Proposed CNN-BiLSTM model demonstrates the highest accuracy among the models analyzed. It outperforms the other models with an accuracy of 97.12%. The CNN-LSTM model follows closely with an accuracy of 96.69%. LeNet5 and AlexNet models achieved accuracies of 93.97% and 89.79% respectively, while the CNN model had the lowest accuracy at 85.94%.
- The recall of the models ranged from 50% to 100%. The proposed CNN-BiLSTM model had the highest recall of 100% for the following classes: U2R, DDoS, DoS, R2L, Botnet, Web Attack, and Normal. The CNN-LSTM model had the highest recall of 90.18% for the class

TABLE 6. Multi classification results on InSDN dataset.

Models	Metrics (%)	Class								
		BFA	U2R	DDoS	DoS	R2L	Botnet	Web Attack	Probe	Normal
CNN	Accuracy	**85.94**								
	Recall	54.74	60.84	85.41	90.25	50	7	16.25	94.86	98.10
	Precision	15.87	50.91	74.15	85.45	18.65	6.36	50	94.69	96.33
	F1-Score	31.65	21.12	76.86	87.61	37.16	5.36	24.52	93.05	94.20
AlexNet	Accuracy	**89.79**								
	Recall	50.54	30.56	80.42	84.25	49.92	48.25	46.25	87.90	97.85
	Precision	48.35	23.85	73.99	88.98	13.43	14.72	10.21	89.85	95.44
	F1-Score	40.54	24.79	75.45	85.47	26.79	29.36	20.37	87.38	97.61
LeNet5	Accuracy	**93.97**								
	Recall	50	50.84	80.64	80.47	92.89	20.38	26.69	99.99	99.71
	Precision	20.35	13.43	83.75	82.25	94.97	26.96	28.36	100	96.58
	F1-Score	40.54	26.79	86.39	85.68	91.98	25.45	29.78	99.98	97.26
CNN-LSTM	Accuracy	**96.69**								
	Recall	90.18	100	99.38	65.12	100	56.73	51.04	98.77	98.45
	Precision	92.42	100	99.78	66	100	51.66	50.88	95.4	97.21
	F1score	91.89	100	98.74	69.37	100	51.53	49.12	93.97	97.51
Proposed CNN-BiLSTM	Accuracy	**97.12**								
	Recall	95.99	100	90	93.75	100	94.87	78.4	99.87	90.18
	Precision	93.43	98.46	85.61	99.03	99.83	96.36	100	100	92.42
	F1-Score	90.58	99.22	87.7	96.18	99.98	95.73	87.89	100	91.89

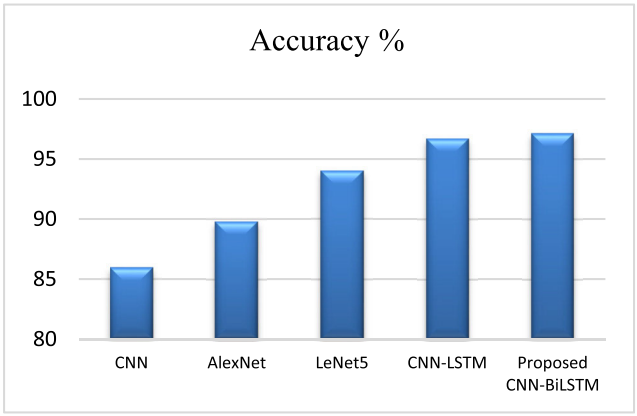

FIGURE 7. Accuracy results of multi class classification on InSDN dataset for different models.

BFA. For the precision of the models ranged from 13.43% to 100%. The Proposed CNN-BiLSTM model had the highest precision of 100% for the class BFA. The CNN-LSTM model had the highest precision of 92.42% for the class U2R.

- Figure 8 shows the F1-score of all models that ranged from 20.35% to 100%. The Proposed CNN-BiLSTM model had the highest F1-score of 100% for the classes U2R, DDoS, DoS, R2L, Botnet, Web Attack, and Normal. The CNN-LSTM model had the highest F1-score of 91.89% for the class BFA.

Overall, the proposed CNN-BiLSTM model had the highest accuracy, recall, precision, and F1-score for most of the classes. This suggests that the Proposed CNN-BiLSTM model is the most accurate model for classifying network attacks.

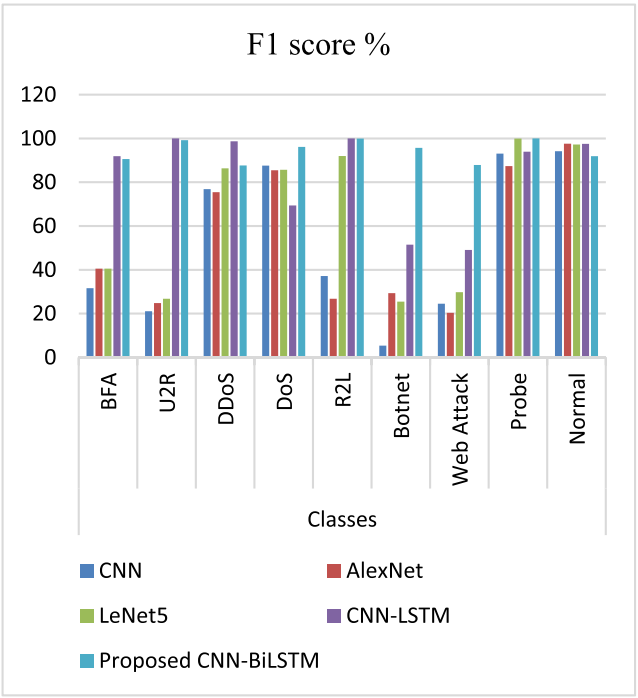

FIGURE 8. All models of multiclass classification on InSDN dataset were scored using the F1 score metric.

2) RESULTS AND DISCUSSION ON NSL KDD DATASET

Table 6 shows the performance of different models on five different classes of network traffic: U2R, R2L, Probe, DoS, and Normal. The models are evaluated using different metrics, including accuracy, recall, precision, and F1-score.

Based on Figure 9 and Table 7, the proposed CNN-BiLSTM model has the highest accuracy of 98.42%. It also has the

TABLE 7. NSL KDD multi class classification results.

Models	Metrics (%)	Class				
		U2R	R2L	Probe	DoS	Normal
CNN	Accuracy	**79.14**				
	Recall	96.32	84.36	86.23	83.07	78.47
	Precision	89.84	88.25	93.34	83.39	52.16
	F1-Score	95.92	91.35	90.54	83.21	68.56
AlexNet	Accuracy	**97.85**				
	Recall	86.67	36.19	97.40	99.75	99.57
	Precision	81.18	38.25	99.10	99.89	96.81
	F1-Score	88.68	53.14	98.28	99.82	98.17
LeNet5	Accuracy	**97.31**				
	Recall	94.86	53.96	96.16	99.44	98.14
	Precision	88.84	94.98	97.13	99.07	98.03
	F1-Score	89.92	68.82	96.64	99.26	98.08
CNN-LSTM		**98.38**				
	Recall	96.67	68.97	97.40	99.89	99.12
	Precision	96.98	94.57	98.74	99.48	98.50
	F1-Score	95.30	79.77	98.07	99.68	98.81
Proposed CNN-BiLSTM	Accuracy	**98.42**				
	Recall	100	74.31	91.05	99.69	99.02
	Precision	99.31	91.39	93.54	99.63	98.35
	F1-Score	99.65	81.97	91.79	99.66	98.69

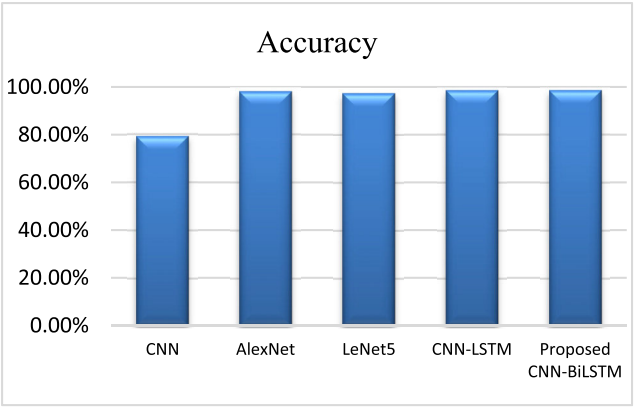

FIGURE 9. Accuracy results of multi class classification on the NSL-KDD dataset for different models.

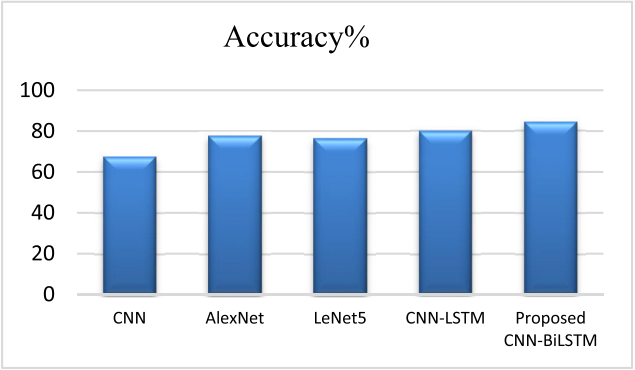

FIGURE 10. Accuracy results of multi class classification on UNSW-NB15 dataset for different models.

highest recall for all five classes of attacks, and the highest precision for three of the five classes. The F1-score for the proposed CNN-BiLSTM model is also the highest for all

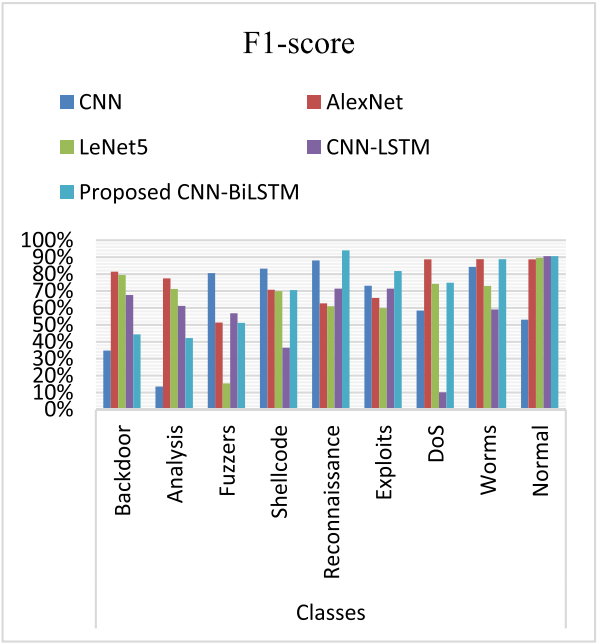

FIGURE 11. All models of multiclass classification on UNSW-NB15 dataset were scored using the F1 score metric.

five classes. The next best model is CNN-LSTM, which has an accuracy of 98.38%. It has the highest recall for the DoS class, and the highest precision for the Probe class. The F1-score for CNN-LSTM is also the highest for the DoS class. The other three models (CNN, AlexNet, and LeNet5) have lower accuracy than the proposed CNN-BiLSTM and CNN-LSTM models. However, they still have relatively high accuracy, with all three models having an accuracy of at least 97%. Overall, the proposed CNN-BiLSTM model is the most accurate model for classifying network traffic. It has

TABLE 8. UNSW-NB15 multi class classification results.

Models	Metrics (%)	Class								
		Backdoor	Analysis	Fuzzers	Shellcode	Reconnaissance	Exploits	DoS	Worms	Normal
CNN	Accuracy	**67.41**								
	Recall	33.45	10.36	50.01	78.65	96.73	92.26	73.08	73.76	45.89
	Precision	26.69	15.6	43.83	65.45	80.77	78.14	48.64	98.32	36.15
	F1-Score	34.8	13.49	80.55	83.24	88.03	73.13	58.41	84.29	53.10
AlexNet	Accuracy	**77.54**								
	Recall	82.36	80.25	66.89	80.76	63.37	88.87	81.64	92.99	87.05
	Precision	76.54	85.73	54.02	67.01	60.79	61.41	77.15	91.34	81.68
	F1-Score	81.39	77.48	51.32	70.7	62.72	65.92	88.69	88.76	88.70
LeNet5	Accuracy	**76.35**								
	Recall	76.59	74.95	47.89	87.48	59.92	70.69	77.99	72.68	87.29
	Precision	74.95	52.21	49.09	66.49	57.67	58.15	71.94	73.46	91.86
	F1-Score	79.45	71.19	15.39	69.86	60.98	59.86	74.25	72.98	89.52
CNN-LSTM	Accuracy	**79.92**								
	Recall	65.25	76.12	55.59	26.06	69.81	88.47	55.00	78.72	91.88
	Precision	63.89	60.05	58.22	61.19	72.98	59.91	51.92	88.50	89.41
	F1-Score	67.58	61.15	56.87	36.55	71.36	71.44	10.08	59.10	90.63
Proposed CNN-BiLSTM	Accuracy	**84.23**								
	Recall	49.29	54.95	68.69	86.08	97.98	93.4	79.31	90.72	91.47
	Precision	49	42.56	54.29	66.83	91.2	75.45	72.31	88.87	92.35
	F1-Score	44.35	42.18	51.16	70.51	94.01	81.82	74.88	88.81	90.58

TABLE 9. Average accuracy of the proposed model compared to other models.

References	Method used	Accuracy for multiclass classification on average		
		InSDN dataset	UNSW-NB15	NSL-KDD
[25]	Deep Neural Network (DNN)	-	-	91.10%
[16]	J48 which is based on decision tree	-	-	75.75%
[9]	CNN-BiLSTM	-	77.16%	83.58%
[42]	GRU-RNN	-	-	89%
[43]	Stacked Auto-Encoders	-	-	94.88%
This study	Proposed CNN-BiLSTM	97.12%	84.23%	98.42%

the highest accuracy, recall, precision, and F1-score for all five classes of network traffic. The CNN-LSTM model is the next best model, with slightly lower accuracy but still very high performance. The other three models (CNN, AlexNet, and LeNet5) have lower accuracy but still relatively high performance.

3) RESULTS AND DISCUSSION ON UNSW-NB15 DATASET

Table 8 shows the performance of different models on an attack detection task. The models are evaluated using different metrics, including accuracy, recall, precision, and F1-score. The proposed CNN-BiLSTM model achieves the highest accuracy of 84.23%. It also has the highest recall and F1-score for most of the classes. This suggests that the proposed model can effectively detect attack with high accuracy.

In Table 8 is a more detailed breakdown of the results for UNSW-NB dataset for the proposed model:

- Figure 10 shown the accuracy of different models, the proposed model has an accuracy of 84.23%, which is higher than all of the other models.
- Recall: The proposed model has a recall of 49.29% for backdoors, 54.95% for analysis, 68.69% for fuzzers, 86.08% for shellcode, 97.98% for reconnaissance, 93.4% for exploits, 79.31% for DoS, and 91.47% for worms. This suggests that the proposed model is capable of efficiently identifying a wide range of attacks.
- Precision: The proposed model has a precision of 49% for backdoors, 42.56% for analysis, 54.29% for fuzzers, 66.83% for shellcode, 91.2% for reconnaissance, 75.45% for exploits, 72.31% for DoS, and 88.87% for worms. This implies that the proposed model is not overfitting the data and is able to generalize to unseen attack samples.
- Figure 11 shown the F1-score of all models for multi class classification on UNSW-NB15, the proposed model has an F1-score of 44.35% for backdoors, 42.18% for analysis, 51.16% for fuzzers, 70.51% for shellcode, 94.01% for reconnaissance, 81.82% for exploits, 74.88% for DoS, and 90.58% for worms. This suggests that the proposed model can effectively detect most types of attack with good accuracy and precision.

Overall, the results show that the proposed CNN-BiLSTM model is a promising approach for attack detection. It can effectively detect most types of attack with high accuracy and precision.

4) COMPARISON OF THE PROPOSED HYBRID MODEL WITH DIFFERENT MODELS

An improved level of accuracy was achieved with the proposed model. Based on Table 9, our model had an accuracy of 98.42%, which was the highest among comparable studies.

VI. CONCLUSION

In this research, we propose a solution for network intrusion detection based on CNN and BiLSTM. The suggested model makes use of the integrity of CNN and BiLSTM to enhance detection capacity across a variety of datasets. Additionally, we demonstrated that our suggested technique outperformed either a single CNN, LeNet5, AlexNet, or CNN-LSTM in terms of training time and accuracy. Furthermore, by balancing the dataset using the random over sampling approach and selecting features using a random forest classifier along with the recursive feature elimination approach, the detection model's performance was improved, indicating that the hybrid model has a significant potential for usage in real-time NIDS. In further work, we attempting to explore the potential of transfer learning techniques to improve the performance of our model. as well as to deploy the suggested model in a real SDN system and evaluate its throughput and latency performance.

ACKNOWLEDGMENT

The authors would like to thank their technical support team for making it possible and for using the TRUBA HPC system efficiently in this article.

REFERENCES

[1] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1955–1980, 4th Quart., 2014, doi: 10.1109/COMST.2014.2320094.

[2] N. McKeown and T. Anderson, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[3] M. Latah and L. Toker, "An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks," *CCF Trans. Netw.*, vol. 3, nos. 3–4, pp. 261–271, Dec. 2020, doi: 10.1007/s42045-020-00040-z.

[4] M. S. ElSayed, N.-A. Le-Khac, M. A. Albahar, and A. Jurcut, "A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique," *J. Netw. Comput. Appl.*, vol. 191, Oct. 2021, Art. no. 103160.

[5] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," *Peer-Peer Netw. Appl.*, vol. 12, no. 2, pp. 493–501, Mar. 2019, doi: 10.1007/s12083-017-0630-0.

[6] M. Kantardzic, *Data Mining: Concepts, Models, Methods, and Algorithms*. Hoboken, NJ, USA: Wiley, 2011.

[7] I. Srba and M. Bieliková, "Encouragement of collaborative learning based on dynamic groups," in *Proc. Eur. Conf. Technol. Enhanced Learn.* Cham, Switzerland: Springer, 2012, pp. 432–437.

[8] M. Abdallah, N. A. Le Khac, H. Jahromi, and A. D. Jurcut, "A hybrid CNN-LSTM based approach for anomaly detection systems in SDNs," in *Proc. 16th Int. Conf. Availability, Rel. Secur.*, Aug. 2021, pp. 1–12, doi: 10.1145/3465481.3469190.

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

[10] N. M. Jacob and M. Y. Wanjala, "A review of intrusion detection systems," *Global J. Comput. Sci. Inf. Technol. Res.*, vol. 5, no. 4, pp. 1–5, 2017.

[11] L. Zhang, J. Huang, Y. Zhang, and G. Zhang, "Intrusion detection model of CNN-BiLSTM algorithm based on mean control," in *Proc. IEEE 11th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Oct. 2020, pp. 22–27.

[12] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020, doi: 10.1109/ACCESS.2020.3022633.

[13] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 1848–1853, 2013.

[14] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," *IEEE Access*, vol. 7, pp. 30373–30385, 2019.

[15] R. K. Malaiya, D. Kwon, J. Kim, S. C. Suh, H. Kim, and I. Kim, "An empirical evaluation of deep learning for network anomaly detection," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Mar. 2018, pp. 893–898.

[16] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2016, pp. 258–263.

[17] S. Boukria and M. Guerroumi, "Intrusion detection system for SDN network using deep learning approach," in *Proc. Int. Conf. Theor. Applicative Aspects Comput. Sci. (ICTAACS)*, vol. 1, Dec. 2019, pp. 1–6.

[18] T. Bakhshi and B. Ghita, "Anomaly detection in encrypted internet traffic using hybrid deep learning," *Secur. Commun. Netw.*, vol. 2021, pp. 1–16, Sep. 2021.

[19] A. R. Narayanadoss, T. Truong-Huu, P. M. Mohan, and M. Gurusamy, "Crossfire attack detection using deep learning in software defined ITS networks," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–6.

[20] M. K. Prasath and B. Perumal, "A meta-heuristic Bayesian network classification for intrusion detection," *Int. J. Netw. Manage.*, vol. 29, no. 3, p. e2047, May 2019.

[21] A. Dawoud, S. Shahristani, and C. Raun, "Deep learning and software-defined networks: Towards secure IoT architecture," *Internet Things*, vols. 3–4, pp. 82–89, Oct. 2018.

[22] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, "Machine-learning based threat-aware system in software defined networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–9.

[23] A. Alshamrani, A. Chowdhary, S. Pisharody, D. Lu, and D. Huang, "A defense system for defeating DDoS attacks in SDN based networks," in *Proc. 15th ACM Int. Symp. Mobility Manage. Wireless Access*, Nov. 2017, pp. 83–92.

[24] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Proc. Int. Symp. Recent Adv. Intrusion Detection*, Menlo Park, CA, USA, Sep. 2011, pp. 161–180.

[25] R. Sathya and R. Thangarajan, "Efficient anomaly detection and mitigation in software defined networking environment," in *Proc. 2nd Int. Conf. Electron. Commun. Syst. (ICECS)*, Feb. 2015, pp. 479–484.

[26] S. T. Selvi and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in *Proc. 6th Int. Conf. Adv. Comput. (ICoAC)*, Dec. 2014, pp. 205–210.

[27] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," 2016, *arXiv:1611.07400*.

[28] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against software defined network controllers," *J. Netw. Syst. Manage.*, vol. 26, no. 3, pp. 573–591, Jul. 2018.

[29] A. Le, P. Dinh, H. Le, and N. C. Tran, "Flexible network-based intrusion detection and prevention system on software-defined networks," in *Proc. Int. Conf. Adv. Comput. Appl. (ACOMP)*, Nov. 2015, pp. 106–111.

[30] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in SDN using machine learning approach," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, Nov. 2016, pp. 167–172.

[31] H. Li, F. Wei, and H. Hu, "Enabling dynamic network access control with anomaly-based IDS and SDN," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, Mar. 2019, pp. 13–16.

[32] P. Manso, J. Moura, and C. Serrão, "SDN-based intrusion detection system for early detection and mitigation of DDoS attacks," *Information*, vol. 10, no. 3, p. 106, Mar. 2019.

[33] M. A. Albahar, "Recurrent neural network model based on a new regularization technique for real-time intrusion detection in SDN environments," *Secur. Commun. Netw.*, vol. 2019, pp. 1–9, Nov. 2019.

[34] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Comput. Secur.*, vol. 86, pp. 147–167, Sep. 2019.

[35] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[36] E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Cham, Switzerland: Springer, 2019.

[37] E. Jackson and R. Agrawal, *Performance Evaluation of Different Feature Encoding Schemes on Cybersecurity Logs*. Piscataway, NJ, USA: IEEE, 2019.

[38] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–45, Jan. 2018.

[39] J. Ling and C. Wu, "Feature selection and deep learning based approach for network intrusion detection," in *Proc. 3rd Int. Conf. Mechatronics Eng. Inf. Technol.*, 2019, pp. 764–770, doi: 10.2991/icmeit-19.2019.122.

[40] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

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

[42] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in SDN-based networks," in *Proc. 4th IEEE Conf. Netw. Softwarization Workshops (NetSoft)*, Jun. 2018, pp. 202–206.

[43] P. Choobdar, M. Naderan, and M. Naderan, "Detection and multi-class classification of intrusion in software defined networks using stacked autoencoders and CICIDS2017 dataset," *Wireless Pers. Commun.*, vol. 123, no. 1, pp. 437–471, Mar. 2022.

RACHID BEN SAID received the M.S. degree in computer engineering from the National High School for Electricity and Mechanics, Hassan II University of Casablanca, Casablanca, Morocco, in 2014. He is currently pursuing the Ph.D. degree in computer engineering with Ankara University, Ankara, Turkey. His research interests include the application of deep learning, and machine learning in the fields of software-defined networking, networking security, and NLP.

ZAKARIA SABIR received the Ph.D. degree in computer engineering and the master's degree in information systems security from the National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra, Morocco. His research interests include connected vehicles, named data networking, security, blockchain, and intelligent transportation systems.

IMAN ASKERZADE received the B.S. and M.S. degrees from the Department of Physics, Moscow State University, Russia, and the Ph.D. degree from the Azerbaijan Academy of Sciences, Moscow State University, Azerbaijan, in 1995. Since 2012, he has been a Professor with the Computer Engineering Department, Ankara University, Ankara, Turkey. His research interests include fuzzy logic, quantum computing, modeling, and simulation.