IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 November 2023, accepted 6 January 2024, date of publication 11 January 2024, date of current version 19 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3353055

RESEARCH ARTICLE

Building a Cloud-IDS by Hybrid Bio-Inspired Feature Selection Algorithms Along With Random Forest Model

MHAMAD BAKRO[1], RAKESH RANJAN KUMAR[1], MOHAMMAD HUSAIN[2], ZUBAIR ASHRAF[3], ARSHAD ALI[2], (Senior Member, IEEE), SYED IRFAN YAQOOB[4], MOHAMMAD NADEEM AHMED[5], AND NIKHAT PARVEEN[6]

[1]Department of Computer Science and Engineering, C. V. Raman Global University, Bhubaneswar, Odisha 752054, India
[2]Department of Computer Science, Faculty of Computer and Information Systems, Islamic University of Madinah, Madinah 42351, Saudi Arabia
[3]Department of Computer Engineering and Applications, GLA University, Mathura, Uttar Pradesh 281406, India
[4]Department of Computer Science and Applications, Dr. Vishwanath Karad MIT World Peace University, Pune 411038, India
[5]Department of Computer Science, King Khalid University, Abha 61421, Saudi Arabia
[6]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh 522302, India

Corresponding author: Mhamad Bakro (mhwb14794@gmail.com)

ABSTRACT The adoption of cloud computing has become increasingly widespread across various domains. However, the inherent security vulnerabilities of cloud computing pose significant risks to its overall safety. Consequently, intrusion detection systems (IDS) play a pivotal role in identifying malicious activities within a cloud system. The considerable volume of network traffic data may contain redundant and irrelevant features that can impact the classification performance of the classifier. In addition, the complexity and time consumption increase while processing such a substantial volume of data in the cloud intrusion detection process. To enhance the performance of the IDS, this study proposes a hybrid feature selection approach, combining two bio-inspired algorithms, namely the grasshopper optimization algorithm (GOA) and the genetic algorithm (GA). The combination of these two algorithms ensures a more efficient search for optimal solutions. A random forest (RF) classifier is trained using those optimal features. Moreover, the proposal addresses the challenge of imbalanced data by employing a hybrid approach: over-sampling the minority classes using an adaptive synthetic (ADASYN) algorithm, while implementing random under-sampling (RUS) for the majority class as needed. This integrated strategy significantly influences each category, enhancing the true positive rate (TPR) while minimizing the false positive rate (FPR), thus improving the overall system performance. The proposed approach was evaluated using three datasets: UNSW-NB15, CIC-DDoS2019, and CIC Bell DNS EXF 2021. The recorded accuracies for these datasets were 98%, 99%, and 92%, respectively. The hybrid feature selection-based IDS demonstrated superior performance in multi-class classification, along with exemplary results for individual classes within the datasets. The proposed strategy exhibited a marked superiority with the random forest classifier, especially when compared to other classifiers including SVM, LR, FLN, LSTM, AlexNet, DNN, DBN, DT, and XGBoost. Moreover, this performance remained consistent and commendable even when benchmarked against contemporary state-of-the-art methodologies across multiple evaluation metrics.

INDEX TERMS Hybrid metaheuristic approach, GOA-GA-based feature selection, UNSW-NB15, CIC-DDoS2019, CIC Bell DNS EXF 2021.

I. INTRODUCTION

Cloud computing (CC) is a model that enables information technology and configurable computing resources such as storage, networks, servers, operating systems, etc. for cloud users at affordable service rates [1]. CC offers its services according to the pay-as-you-use policy [2]. Cloud computing has three main service models: software-as-a-service (SaaS), infrastructure-as-a-service (IaaS), and platform-as-a-service (PaaS) [3]. The characteristics of the cloud, such

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

as the elimination of expensive infrastructure, minimized data management costs, and usage-based service charges, reduce the financial loss to users/organizations [4]. Although cloud computing has several advantages like scalability, fast deployment, rapid elasticity, etc. It also faces serious disadvantages in terms of security. Recently, several reports of security breaches in cloud computing have been reported. In 2022, the global cybersecurity market was valued at USD 153.65 billion. The market is expected to increase at a compound annual growth rate (CAGR) of 13.8% during the forecast period, from USD 172.32 billion in 2023 to USD 424.97 billion in 2030 [5]. As the network functions as the lifeblood of the cloud and enables the delivery of cloud services to customers, any risks or imperfections present within it promptly impact the overall security and growth of the cloud. Consequently, fortifying the network against any possible breaches is of crucial importance. The detection and prevention of network threats are the primary security challenges in cloud computing. Due to inadequate defensive measures, network intrusions have recently increased; the IDS can be deployed to tackle these security issues. Intrusion detection systems are critical tools for cybersecurity, especially in identifying attacks in cloud computing environments. Traditional solutions handle intrusions with firewalls, attack-resistant software, access control, encryption, etc. However, these techniques were suitable only for small-scale attack identification. Conventional approaches are infeasible with a large number of modern attacks [6]. Cloud computing is susceptible to various types of modern attacks, including but not limited to denial-of-service (DOS), distributed denial-of-service (DDOS), and domain name system (DNS) attacks [7]. With the increase in applications and devices on the Internet, automatic intrusion detection plays a major role since manual approaches are unable to handle a large amount of data [8]. Nevertheless, before integrating any IDS model into the cloud ecosystem, it's essential to ensure its robustness and functionality. Thus, the objective of this study is to build such an effective IDS.

In summary, the primary motivation for this research lies in addressing the critical challenge of intrusion detection in network-driven systems, including cloud computing. Such systems are frequently subjected to similar types of attacks. For instance, the notable increase in the adoption of cloud-based services has precipitated a parallel escalation in security threats, highlighting the need for effective intrusion detection systems. Traditional IDS approaches often prove inadequate, primarily due to their limited capacity to manage the dynamic and intricate nature of cloud services, as well as the difficulties arising from imbalanced datasets, which are characterized by high dimensionality and numerous superfluous features. This scenario intensifies the need for advanced, AI-enhanced IDS techniques, which are the focus of our research.

IDS based on AI utilize ML and DL models to construct an efficient system to shield the cloud from attacks [9]. During the intrusion detection process, ML models encounter certain limitations and their performance deteriorates when the complexity of attacks increases [10]. IDS approaches that are built with DL algorithms, such as ANN, RNN, CNN, etc., have demonstrated the ability to process large volumes of high-dimensional network traffic with enhanced accuracy compared to ML-based IDS. However, these approaches consume an excessive amount of resources, either in terms of time or RAM. Each approach has its own strengths and weaknesses. It's important to note that ML models generally consume fewer resources than DL models. However, in some instances, DL models exhibit superior performance. This underscores the significance of ensemble learning (EL) models, which strike a balance between impressive performance and reasonable resource consumption. RF model is among the most renowned of these due to its structure and its method for obtaining optimal solutions.

The authors in references [11] and [12] found that the ML and DL strategies employed in IDS failed to effectively handle multi-classification due to the high dimensionality and non-linear characteristics of the data. Feature selection mitigates this issue by eliminating unimportant features, thereby enhancing the learning process. Irrelevant features can lead to overfitting and bias the solution, which in turn, negatively affects the performance of the classification process. Additionally, these features consume a substantial amount of computational resources and increase processing time. Optimal feature selection can relieve these issues by conserving resources. Furthermore, when a classifier is trained with these optimal features, it not only improves classification performance but also expedites both the training and testing phases [13]. Bio-inspired algorithms used in IDS offer advantages such as a reduction in feature sets and the selection and ranking of discriminative features. By reducing feature sets with bio-inspired algorithms, we can achieve superior predictive models with less computational complexity and cost during the training and testing of a classifier [14]. It is observed from previous research that the selection of relevant features plays a major role in achieving higher prediction accuracy and reducing computational costs. Therefore, we have focused on improving the feature selection process through a hybrid optimization algorithm in which GOA and GA are combined to obtain optimized features [15].

In this research, the original data are collected from the UNSW NB-15, CIC DDOS 2019, and CIC Bell DNS 2021 datasets. Subsequently, data pre-processing is conducted with label encoding, min-max scaling, and a combination of the ADASYN and RUS methods for better data representation. The proposed IDS mechanism involves two stages, namely feature selection and classification. The feature selection process involves hybridizing the GOA and GA with the aim of selecting optimal features. RF model is chosen to classify the attacks and will be trained using these optimally selected features. The performance of the proposed method is compared with existing approaches in the context of multi-class classification. The major contributions of the research are:

1. Imbalanced Data Handling: Our methodology uniquely addresses the imbalanced data issue using a combination of over-sampling for minority classes and under-sampling for majority classes, by employing the ADASYN and RUS techniques. This not only aids in mitigating overfitting and underfitting but also significantly enhances the performance of each category in the dataset. This approach is specifically tailored to manage the challenges posed by uneven data distributions, which are common in intrusion detection scenarios.
2. Hybrid Feature Selection Approach: The integration of GOA and GA for feature selection in our study represents a novel approach, combining their exploration and exploitation abilities. This hybrid method leverages the strengths of both algorithms, ensuring a more efficient and comprehensive feature selection process. This not only reduces the complexity of the model but also contributes to a reduction in overfitting, thereby enhancing the overall performance of the classifier.
3. RF Classifier Efficiency in Training: The utilization of optimally selected features in training the Random Forest classifier underscores the methodological advancement of our study. RF, an ensemble learning-based model, has shown remarkable effectiveness owing to its structure and notable attributes. These include the ability to manage both categorical and continuous data, handle missing values and outliers, and the requirement of less training time, thereby making it our classifier of choice.
4. Utilization of Latest Datasets: Our methodology is rigorously assessed using the latest and most comprehensive datasets, namely UNSW-NB15, CIC-DDoS2019, and CIC Bell DNS 2021. This enhances the relevance and applicability of our findings. These datasets encompass a broad spectrum of threats, accurately representing prevalent real-world attacks. The CIC Bell DNS 2021 dataset is regarded as the most recent and has not been extensively utilized for evaluations in traditional studies.
5. Robustness and Generalizability: The exceptional performance of our methodology across multiple datasets, as demonstrated in our study, signifies its robustness and low variance. This suggests a strong capability of our approach to generalize across various real-world scenarios, representing a significant improvement over existing methodologies. Unlike ours, these methodologies might not exhibit such versatility, often being tested using only one or two outdated datasets.

This study is organized as follows: Section II reviews the cutting-edge works related to IDSs. Section III provides a brief overview of the proposed system and its steps. In Section IV, we delve into the simulation process and discuss the results. Section V details the challenges and limitations of the approach. Section VI delineates key considerations that will facilitate the practical implementation of our approach. Finally, section VII presents the conclusion and future work.

II. RELATED WORKS

This section offers a concise overview of state-of-the-art studies that have sought to improve IDS performance through the application of feature selection methods such as metaheuristic-inspired and other related approaches, as well as ML, DL, and EL models. Rashid et al. [6] suggested a tree-based stacking ensemble technique (SET) model with the objective of selecting the best features. The feature selection approach reduces the dimensionality of network data. The stacking ensemble model is constructed using a decision tree (DT), a random forest (RF), and XGBoost for classification. The proposed IDS model achieved 99.9% and 95.26% accuracy on the NSL-KDD and UNSW-NB15 datasets, respectively. Ghosh et al. [8] introduced an efficient IDS designed using the dolphin mating model for feature selection. The dolphin mating algorithm in the proposed research removes irrelevant features to achieve higher accuracy and reduce the computational cost. Kanna and Santhi [9] suggested a hybrid IDS model in order to handle the massive volume of network traffic data. The classification is performed using a MapReduce-based deep learning model called the convolutional-long short-term memory network (CONV-LSTM). Initially, the feature selection is carried out using the artificial bee colony (ABC) algorithm. The hyperparameters of the CONV-LSTM networks are optimized by the black widow optimization (BWO) algorithm. The proposed approach achieves an accuracy of 98.67%, 97.003%, 98.667%, and 98.25% on the NSL-KDD, ISCX-IDS, UNSW-NB15, and CSE-CIC-IDS2018 datasets, respectively. Nevertheless, the duration of training (26,721.2 seconds) and testing (402.67 seconds) is still quite lengthy, indicating substantial usage of resources.

Cai et al. [10] developed an efficient IDS employing hybrid parallel deep learning, which comprises two parallel CNN models that combine spatial features obtained from full convolution. The spatio-temporal features are classified using the CosMargin classifier. The approach was assessed on two datasets, namely CIC-IDS 2017 and ISCX 2012, and it achieved a 99% accuracy rate in detecting the malicious class. Binbusayyis and Vaiyapuri [11] proposed an IDS approach that utilizes a combination of convolutional autoencoder (CAE) and one-class SVM (OCSVM). The approach leverages 1D CAE for feature representation and OCSVM for classification. The objective function is designed by combining the reconstruction and classification error. The proposed approach was evaluated on the NSL-KDD and UNSW-NB15 datasets and achieved better results than conventional approaches. Kannari et al. [12] developed an IDS strategy using a sparse autoencoder with a swish-PReLU activation function, which enhances classification accuracy and speed. The optimal feature sets are selected with two approaches, namely the second percentile method and recursive feature elimination. The proposed IDS mechanism is simulated with the NSL-KDD, CIC-IDS2017, and AWID databases. The approach showed an improvement of more than 4.77% compared to traditional strategies. Dora and Lakshmi [16] presented a DDOS detection scheme that uses a fusion of CNN and optimized LSTM. Feature selection is carried out using the closest position-based grey wolf optimization (CP-GWO) technique, while CNN is used for feature learning. LSTM is optimized with CP-GWO to enhance accuracy by optimizing the neurons. Although the proposed DDoS detection model showed enhanced accuracy compared to conventional schemes, the study failed to discuss the time consumption of the proposed method. Kanna and Santhi [17] proposed a model called OCNN-HMLSTM, which integrates optimized CNN (OCNN) to extract spatial features which used lion swarm optimization (LSO) to tune hyperparameters and a hierarchical multi-scale LSTM (HMLSTM) that effectively handles temporal attributes and classifies network packets. This system is effective for identifying threats because it can automatically interpret spatial-temporal features. However, the use of deep learning methods results in a complex model that requires a long training duration (30,665 s) and consumes significant resources. Authors plan to explore feature selection methods in the future. Kim and Pak [18] presented a real-time intrusion detection approach that combines deferred decision and hybrid classification techniques. A packet- and session-based feature were fused to create a hybrid ML model to achieve intrusion detection without delay. However, due to the complex nature of the proposed approach, it may only be suitable for high-level networks.

Dwivedi et al. [19] suggested an approach for feature selection using ensemble feature selection (EFS) and grasshopper optimization algorithm (GOA) to enhance the performance of the support vector machine (SVM) classifier. The EFS method was used to rank the features, and GOA was employed to select the most relevant features. The approach showed significant improvement in the DR while reducing the FAR for the NSL-KDD and KDD Cup 99 datasets. Ahmet Ali Suzen [20] presented a hybrid deep belief network (DBN) to detect intrusions in network traffic. The DBN is updated with contrastive divergence. The proposed DBN model achieved 99.72% accuracy in classification using a dataset produced by ICS. Sajith and Nagarajan [21] introduced an IDS that employs deep learning and a metaheuristic algorithm. The feature selection process is performed with particle swarm optimization (PSO), while the classification is executed with a DBN model. The proposed approach is tested on the DARP 1999 dataset, and the results show that it outperforms conventional methods with an accuracy of 96.5%. Sreelatha et al. [22] developed a cloud-based IDS that efficiently selects relevant features from the intrusion dataset using the sandpiper optimization algorithm (SOA). The EEDTL technique is then utilized to classify samples using the selected features. The EEDTL approach uses a pre-trained network, AlexNet, to effectively configure the convolution layer features. The network weights are updated using the extended equilibrium optimizer (EEO). The proposed approach is tested using the UNSW-NB15 and NSL-KDD datasets. Liu et al. [23] presented a deep learning approach for intrusion detection that is based on multi-task learning. First, the authors visually analyzed the datasets to study the imbalanced distribution and clustering features of each attack. Then, they developed a framework to tackle the class imbalance issue. The auto-encoder-based contrastive learning, supervised learning-based clustering model, and MLP-based classifier are combined to develop a single framework to detect the intrusion. The focal loss is used to solve the problem of imbalanced binary classification. The suggested methodology was evaluated using the NSL-KDD, AWID, and Bot-IoT datasets. Moizuddin and Jose [24] introduced a new approach to detecting intrusions in the network by proposing a generalized mean grey wolf algorithm (GWGWO) and an ElasticNet contractive autoencoder (ECAE). In this approach, feature selection is carried out using GWGWO, which selects the most important features from the given dataset. The selected features are then trained using stacked ECAE. The classifier model performed both binary and multi-class classification with an accuracy of 99% on the NSL-KDD and BoT-IoT datasets. Hsu et al. [25] suggested an IDS that uses a combination of LSTM and CNN. The proposal employed deep learning models to solve the shortcomings of ML-based approaches in IDS. The proposed system uses CNN to extract relevant features and LSTM for classification. The system was simulated using NSL-KDD datasets, and it outperformed other conventional works. Ghosh et al. [26] proposed a method for intrusion detection in cloud environments using the sage-grouse mating algorithm (SGM). SGM is utilized for feature selection and to reduce redundant data. The proposal was tested on two datasets, NSLKDD and Kyoto2006+, and the performance of the algorithm was evaluated using ML classifiers such as neural networks (NN), KNN, Bagging, DT, and RF to classify the attack type.

Qazi et al. [27] suggested a DL-based IDS that uses a combination of the stacked non-symmetric deep autoencoder as well as the SVM for classification. The proposed approach was verified using the KDD Cup 99 dataset and showed improved accuracy compared to traditional approaches. Gupta et al. [28] presented a hybrid IDS that combines optimization techniques and DL to identify intrusions. The hybrid chicken swarm genetic algorithm (HCSGA) and min k-means algorithm perform feature selection and clustering. The proposed IDS then classifies the network traffic as either attack or non-attack using a deep learning hybrid neural network algorithm (DLHNN). The DLHNN is an enhanced version of the ANN, which is known to suffer from issues related to backpropagation, leading to longer training times. To address this issue, the levy-flight centered crow search algorithm (LF-CSA) is combined with the ANN to optimize the network weights and avoid the backpropagation problem. The experimental results showed that the proposal achieved a high accuracy of 99.52% and a low training time of only 362 seconds on the NSL-KDD dataset. Balamurugan et al. [29] developed an IDS that incorporates game theory (GT) and a deep neural network (DNN). The GT is integrated into the DNN network to select optimal solutions and improve the classification performance of DNN. The CIC IDS 2017 dataset was used to assess the proposed system, which demonstrated higher accuracy, precision, and detection rate than conventional strategies. Du et al. [30] designed a cloud-fog-IDS based on SVM. To tackle the higher dimensionality of the dataset, the principal component analysis (PCA) technique is employed to reduce its dimensions. Additionally, PSO is utilized to optimize the SVM for training purposes. The proposed IDS scheme was verified with the KDD CUP 99 dataset, achieved better performance, and addressed the issues of storage and calculation overhead. Rehman et al. [31] focused on detecting DDOS attacks using gated recurrent units (GRU). To evaluate the effectiveness of the proposed IDS scheme, the CICDDoS2019 dataset was used. The research employed different classifiers such as GRU, recurrent neural networks (RNN), naive bayes (NB), and sequential minimal optimization (SMO). The results indicated that the GRU algorithm exceeded all other classifiers, achieving the highest accuracy. Wei et al. [32] introduced a hybrid model for detecting DDOS attacks, which involves using an autoencoder and a multi-layer perceptron network (AE-MLP). The autoencoder is used to select relevant features, and the MLP network is used to classify attacks into different DDOS attack types. The approach was estimated via the CCIDDoS2019 dataset. The proposed scheme overcomes the issues of performance overhead and bias caused by the massive volume of datasets.

Patil et al. [33] proposed a blueprint for a hypervisor-level distributed network security (HLDNS) system. This system is integrated into each processing server within a cloud-based settings. A dedicated server, tasked with intrusion detection, monitors network activities between both the internal and external networks linked to the underlying virtual machines (VMs). By employing a combination of rule-based and misuse-based detection techniques, the model is capable of identifying both known and unknown attacks. Moreover, they implemented misuse-based detection prior to rule detection, effectively reducing the overall computational costs as the network data only needed to be examined for zero-day threats. To identify useful features from the cloud network traffic, they enhanced the BBA by introducing two new fitness functions. The features engineered through this method were subsequently utilized in the RF model for intrusion detection. Finally, the misuse (signature) database was updated using attack data gathered from several servers. Bakro et al. [34] offered an effective IDS that incorporates a feature selection concept comprised of IG, CS, and PSO to select meaningful features. Additionally, the system addresses data imbalance issues using SMOTE and employs a random forest for classification. This approach was tested on the UNSW-NB15 and Kyoto datasets. Amerah [35] proposed a network intrusion detection and prevention system (NIDPS) framework, which commences with the normalization of the dataset, using the standard deviation and mean of feature columns. Subsequently, an improved salp swarm algorithm (ISSA) is utilized for automated feature selection. The selected features are then subjected to the SMOTE–Tomek method of over-sampling and under-sampling for class balancing. Four ML classifiers, namely, RF, Extra Tree, DT, and gradient descent boost, are employed, with the results indicating that RF outperforms the others. The proposal was evaluated using the unbalanced UNSW-NB15 dataset for binary and multi-class classification of network attacks.

After reviewing previous research and examining Tables 10, 11, and 12, we discovered that many studies used metaheuristic-inspired algorithms to select features and/or optimize classifiers, while others utilized alternative methods like a filter or wrapper methods for feature selection. Most previous studies utilized tree-based classifiers, such as XGBoost and RF. They also used neural network (NN)-based classifiers, which include models from both ML-NN like FLN and ELM, and DL-NN such as LSTM. These often outperform other models. However, due to the structure of NN models, they consume extra resources. Therefore, tree-based classifiers are considered superior in many scenarios. Additionally, certain approaches had constraints, such as inadequate data processing and feature handling, neglecting to address data imbalances, or using small or outdated datasets. Also, some methodologies were complex and their results were difficult to interpret.

To design an effective intrusion detection system, meticulous preparation of the dataset is crucial. In this respect, we propose addressing data imbalance using a combination of ADASYN and RUS, followed by the application of feature selection techniques to reduce dataset size and enhance performance. We employ bio-inspired algorithms, which are instrumental in selecting the optimal set of features, and have found commendable results using these methods. For classification, we have found that tree-based models perform exceptionally well; consequently, we have used the RF model due to its numerous advantages. Furthermore, we utilize the most recent datasets, which cover a wide range of significant potential zero-day attacks, to evaluate the performance of our system.

III. METHODOLOGY

Before we can implement an IDS in a cloud environment, it is paramount to preprocess the datasets. These datasets encapsulate a variety of attacks as well as a vast amount of irrelevant information. Therefore, it is necessary to identify and isolate the most pertinent features for the training process. The selected features will then be used to train a classifier capable of differentiating between normal packets and various forms of attacks. Consequently, in this section, we introduce our proposed IDS which consists of four basic phases: data preprocessing, feature selection utilizing the GOA-GA, addressing the imbalance in the training set, and implementing the RF model for result generation. Each step will be discussed in detail in the subsequent sections. Figure 1 provides an overview of our proposed approach.

A. DATASETS

The creation of an exhaustive dataset is an expensive undertaking, necessitating significant financial investment and advanced knowledge. Thus, a key challenge faced by IDSs involves the methodical development of a comprehensive, state-of-the-art dataset. This dataset must encapsulate a wide array of contemporary threats categories and accurately reflect the real-world settings. Datasets such as UNSW-NB15, CIC-DDoS2019, and CIC Bell DNS EXF 2021, which are acknowledged as labeled network traffic datasets, are suitable for evaluating intrusion detection systems. The statistical information pertaining to the datasets utilized in this work is presented in Table 1.

TABLE 1. Statistics of utilized intrusion datasets.

Name	No of Attributes	No of Samples	No of Classes	Source
UNSW-NB15	49	500,000	10	[36]
CIC-DDoS2019	88	550,000	13	[37]
CIC Bell DNS EXF 2021	43	711,245	4	[38]

To bolster the research process, it is imperative that the dataset undergoes regular updates to incorporate the most up-to-date prevalent attack scenarios. While the three previously referenced datasets in the study may not include all potential cases in network traffic, they are acknowledged to encompass the most critical recent attack types. A concise review of each of these datasets is provided below:

1) UNSW-NB15

The dataset created by Moustafa and his team [39]. It was designed to handle the challenges inherent in the KDD-Cup 99 and NSLKDD datasets [40]. These two datasets are widely used in the field of intrusion detection research as benchmarks for evaluating novel systems. Compiling this dataset was an intensive process, requiring a diverse range of tools and a substantial time investment of 31 hours to accumulate 100 GB of data, which encompasses approximately 2.5 million samples [41]. The dataset comprises 49 features, gathered utilizing Bro-IDS, Argus tools, and 12 innovative algorithms. Every attribute, along with its associated data type, is displayed in Table 21. Of these features, the majority (43) are numerical, with the remaining six being nominal. These attributes are organized into five distinct groups: basic, time, flow, and content features as well as additional features. Within the dataset, there are two unique features functioning as labels: "attack_cat" and "label". "Attack_cat" encompasses ten classes, including 'dos', 'fuzzers', 'exploits', 'generic', 'reconnaissance', 'backdoor', 'analysis', 'worms', 'shellcode', and 'normal'. While the "label" feature differentiates between an attack (1) and normal traffic (0). In total, the dataset distinguishes between nine types of attacks, which represent a significant variety of threats that any networked system may encounter, along with 'normal' traffic indicating the absence of an attack. In this study, our main focus is on the "attack_cat" feature, which serves as the target variable in our multi-class classification scenario. Due to the substantial size of the dataset, only a selected subset is utilized for evaluation [42]. Table 2 provides a detailed division of the number of instances utilized for each category during training and testing, along with the new training set which is generated after balancing.

TABLE 2. Subset of the UNSW-NB15 dataset utilized in the study.

No	Class Name	Used Dataset	Training Part	New Training Set	Testing Part
1	Normal	436,439	327,422	261,938	109,017
2	DoS	3170	2360	2360	810
3	Fuzzers	4701	3561	3561	1140
4	Exploits	8950	6739	6739	2211
5	Generic	42,610	31,815	31,815	10,795
6	Reconnaissance	2818	2101	2101	717
7	Backdoor	455	342	2906	113
8	Analysis	525	404	2959	121
9	Worms	37	27	2994	10
10	Shellcode	295	229	2965	66
Total		**500,000**	**375,000**	**320,338**	**125,000**

2) CIC-DDoS2019

DDoS attack is a well-known threat to network security. This type of attack seeks to overload target networks with malicious traffic until the network's available bandwidth is fully consumed. This excessive traffic can lead to a significant decrease in system performance and, in extreme cases, precipitate a complete system crash. The CICDDoS2019 dataset, developed by Sharafaldin et al. [43] at the Canadian Institute for Cybersecurity, was created specifically to train models for the detection of various types of DDoS attacks. The CICDDoS2019 dataset remedies the limitations found in other datasets. It encompasses both benign data and the most recent common DDoS attacks, which can be conducted at the application layer through TCP/UDP-based protocols, thus accurately reflecting authentic real-world data. These attacks can be classified into two main categories: reflection-based and exploitation-based [44]. Both categories aim to obscure the attacker's identity and direct traffic to reflector servers, utilizing the target's IP address as the source IP. This strategy results in the inundation of the target system with an overwhelming volume of response packets. The reflection-based category includes such attacks as TFTP, LDAP, SNMP, SSDP, NETBIOS, MSSQL, DNS, and NTP. In contrast, the exploitation-based category is composed of SYN, UDP, UDP-Lag, and WebDDoS. This brings the total to twelve distinct classes of attacks, characterized by 88 features [45]. Each of these attacks is explained in reference [46]. Table 22 presents every attribute alongside its associated data type, highlighting that of the total attributes, 82 are

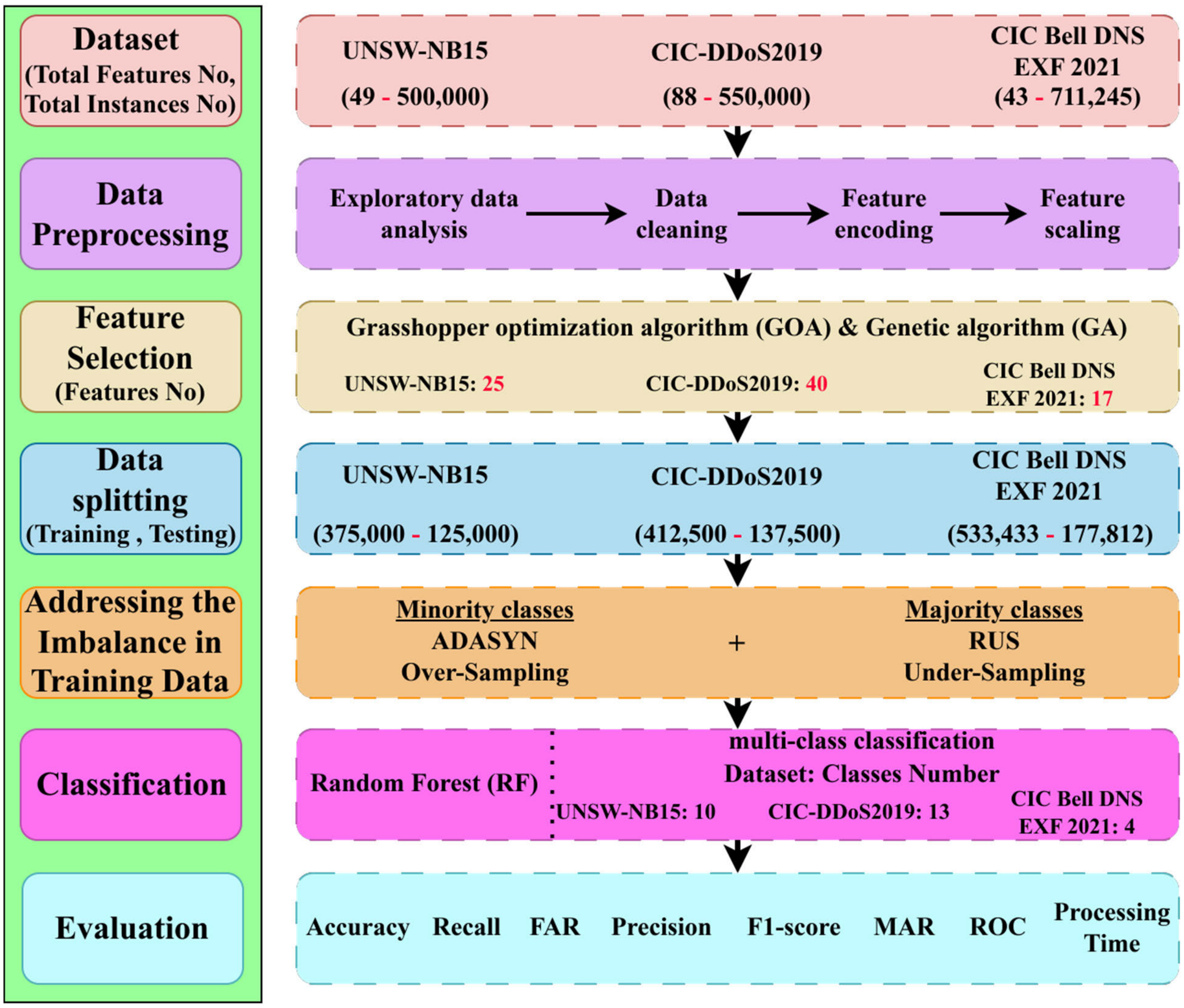

Dataset (Total Features No, Total Instances No)
UNSW-NB15 (49 - 500,000)
CIC-DDoS2019 (88 - 550,000)
CIC Bell DNS EXF 2021 (43 - 711,245)
Data Preprocessing
Exploratory data analysis
Data cleaning
Feature encoding
Feature scaling
Feature Selection (Features No)
Grasshopper optimization algorithm (GOA) & Genetic algorithm (GA)
UNSW-NB15: 25
CIC-DDoS2019: 40
CIC Bell DNS EXF 2021: 17
Data splitting (Training , Testing)
UNSW-NB15 (375,000 - 125,000)
CIC-DDoS2019 (412,500 - 137,500)
CIC Bell DNS EXF 2021 (533,433 - 177,812)
Addressing the Imbalance in Training Data
Minority classes ADASYN Over-Sampling
+
Majority classes RUS Under-Sampling
Classification
Random Forest (RF)
multi-class classification
Dataset: Classes Number
UNSW-NB15: 10
CIC-DDoS2019: 13
CIC Bell DNS EXF 2021: 4
Evaluation
Accuracy
Recall
FAR
Precision
F1-score
MAR
ROC
Processing Time

FIGURE 1. The block diagram of the proposed IDS based on a hybrid of bio-inspired algorithms.

digital, and 6 are categorical. The 'Label' attribute serves as the target variable. Table 3 presents a detailed distribution of the number of instances employed for each category during the training and testing phases, in addition to showcasing the new training set generated after the balancing process.

3) CIC BELL DNS EXF 2021

The Domain Name System (DNS) exfiltration is a technique in which compromised devices transmit encoded data to an attacker's server via DNS request messages [47]. This method is frequently utilized to extract sensitive information from target networks and to establish clandestine communication channels [48]. Despite the implementation of firewalls to scrutinize DNS traffic, adversaries often succeed in conveying encoded data to compromised servers under their control. To develop our detection mechanism for such attacks, we utilized the most recent dataset, CIC Bell DNS EXF 2021, which encompasses a significant number of instances of these attacks [49]. This will serve to enhance the precision and effectiveness of our proposal. This dataset, released by Mahdavifar et al. [50], comprises 270.8 MB of DNS traffic, generated through the exfiltration of various file types of diverse sizes. It incorporates 42 features, extracted from the DNS packets, amounting to approximately 1,019,318 samples. Out of these 42 features, 15 are stateless, 26 are stateful, and 'timestamp' feature. Stateless features, extracted from individual DNS query packets, are independent of the temporal characteristics of queried domains or the DNS activity of hosts, thus minimizing computational strain during real-time operations. In contrast, stateful features consider a sequence of queries within a specific time window, imposing a greater computational burden on the detection system. However, the advantage of stateful detection lies in its capacity to scrutinize DNS logs over an extended timeframe, thus equipping it to handle low-intensity and prolonged DNS attacks. In our study, for a more robust and comprehensive defence against data exfiltration over DNS, we employ a combination of stateless and stateful features. This hybrid approach can facilitate swift threat detection (due to stateless features) while also empowering the system to identify more complex, slow-developing anomalies (attributable to stateful features). The samples are categorized into four classes: 'heavy' and 'light' attacks, as well as 'heavy' and 'light' benign [38]. In relation to these classes, we have introduced an additional feature, 'label.' This feature serves as the target variable, representing the type of instance, bringing the total count of attributes to 43. Each 'heavy' and 'light' attack category encompasses six file types, namely, compressed, audio, image,.exe, video, and text. The features and their corresponding data types are elucidated in Table 23. Among the 43 features, 11 are nominal and 32 are numerical. Table 4 offers a comprehensive delineation of the number of samples utilized for each class during both the training and testing phases. Additionally, it reveals the new training set derived from the balancing process.

TABLE 3. Subset of the CIC-DDoS2019 dataset utilized for evaluation.

No	Class Name		Used Dataset	Training Part	New Training Set	Testing Part
1	BENIGN		16,648	12,448	12,448	4200
2	Reflection	TFTP	49,995	37,503	37,503	12,492
3		LDAP	49,993	37,466	37,466	12,527
4		SNMP	49,957	37,543	37,543	12,414
5		SSDP	49,891	37,415	37,415	12,476
6		NetBIOS	49,839	37,242	37,242	12,597
7		MSSQL	49,352	37,232	37,232	12,120
8		DNS	48,298	36,255	36,255	12,043
9		NTP	37,653	28,215	28,215	9438
10	Exploitation	Syn	49,995	37,532	37,532	12,463
11		UDP	49,990	37,422	37,422	12,568
12		UDP-lag	48,247	36,119	36,119	12,128
13		WebDDoS	142	108	2987	34
Total			**550,000**	**412,500**	**415,379**	**137,500**

TABLE 4. Subset of the used CIC Bell DNS EXF 2021 dataset.

No	Class Name	Used Dataset	Training Part	New Training Set	Testing Part
1	Heavy-Benign	250,710	188,060	150,448	62,650
2	Light-Benign	82,859	62,128	62,128	20,731
3	Heavy Attack	323,698	242,810	194,248	80,888
4	Light Attack	53,978	40,435	40,435	13,543
Total		**711,245**	**533,433**	**447,259**	**177,812**

B. DATA PREPROCESSING

The datasets encompass a wide array of nominal and digital data, among others, including instances of noise, missing values, and unusable data formats. However, some data points may display skewed or irregular values, potentially leading to suboptimal results and negatively impacting the comprehension and performance of machine learning models. This section delves into data pre-processing, an essential step for preparing data for feature selection, given that machine learning algorithms require numerical and cleansed inputs. The following steps are undertaken to achieve this:

1) EXPLORATORY DATA ANALYSIS (EDA)

It is a pivotal procedure enabling a thorough understanding of key attributes of input data elements, such as instances (rows) and features (columns), utilizing statistical summaries and visualization methods based on Python libraries. This process produces highly effective results by ensuring accurate execution of subsequent data processing stages. Post EDA, we receive a comprehensive overview of the features and their data types. This process also aids in identifying missing values, zeros, 'inf,' negative values, duplicate data, outliers, and class imbalances, in addition to determining the total sample count for each class within the datasets, among other insights.

2) DATA CLEANING

It is the process of identifying and rectifying or removing erroneous, incomplete, irrelevant, or missing data, such as replacing missing or NaN values with the mean of the corresponding attribute. A significant portion of the effort is dedicated to ensuring the input data is clean and free of errors, as poor-quality data can result in biased outcomes, lower accuracy, and higher error rates in a model [45].

3) FEATURE ENCODING

The input datasets comprise various columns with a mix of numerical and categorical values. Feature encoding is the process of converting non-numerical data into a numerical format. Most features in this study are numerical, while the rest are nominal. As machine learning models interpret numerical inputs, it becomes necessary to convert these categorical features into a machine-readable format. One-hot encoding and label encoding are common methods for this conversion, each with its own pros and cons. Although one-hot encoding may provide superior performance, it significantly increases the dimensionality of the features [51]. For this study, label encoding was chosen due to its satisfactory results [6]. Integrating one-hot encoding with label encoding would amplify the feature count and enlarge the dataset, resulting in higher resource usage. Since the objective is to reduce the number of features, utilizing label encoding alone—assigning a unique number, commencing from 0, to each unique text of a feature—proves to be more efficient.

4) FEATURE SCALING

The values of numerous features in the datasets span different units and magnitudes. To enhance the efficiency of machine learning methods, it's advantageous to scale input attribute values to a uniform range. Consequently, prior to model training, we normalize and convert all attribute values to fit within a specified range. This crucial step reduces any bias due to features with larger values. Significant techniques for feature scaling encompass standardization (commonly referred to as the Z-score method) and normalization (also referred to as min-max scaling), which frequently yield satisfactory findings [41]. Therefore in this study, we opted for the min-max method, elaborated as below:

$$\mathrm{B} = \frac{\mathrm{B}-\mathrm{B}_{\min}}{\mathrm{B}_{\max}-\mathrm{B}_{\min}} \tag{1}$$

In this context, B_{min} and B_{max} represent the minimum and maximum values of the feature B, respectively. In subsequent step C, the ideal attributes will be determined from the pre-processed dataset.

C. FEATURE SELECTION

The objective of feature selection is to enhance classification performance by selecting the most optimal set of features [52]. A trimmed dataset provides faster training times and greater accuracy. Accordingly, we propose utilizing a fitness function that fulfills objectives like maximizing classification accuracy and minimizing features. In this context, we will briefly discuss our feature selection approach using the Genetic-Grasshopper optimization algorithm.

$$\mathrm{F} = \max\left(\mathrm{Acc}+\mathrm{w_f}\left(1-\frac{\mathrm{F_s}}{\mathrm{F_t}}\right)\right) \tag{2}$$

where Acc represents the accuracy, $\mathrm{w_f}$ is the weight factor, F_s is the length of selected features, and $\mathrm{F_t}$ is the total number of features. The value of the weight factor is set close to one means that both goals—accuracy enhancement and feature minimization—are considered equally important [53].

1) GRASSHOPPER OPTIMIZATION ALGORITHM (GOA)

GOA is a bio-inspired algorithm that emulates the foraging behaviour of grasshopper swarms. Grasshoppers, notorious for their potential to inflict severe damage to crops, undergo a life cycle comprising distinct stages: egg, nymph, and adult. From eggs emerge nymphs, which are small, wingless, and capable of only slow, short-distance movement, characterized by a small step size. These nymphs gradually metamorphose into adult grasshoppers, capable of traversing long distances. These two movements, corresponding to the nymph and adult phases, respectively, represent exploitation and exploration [54]. The mathematical model of the GOA is expressed as follows:

$$X_i = S_i + G_i + W_i \tag{3}$$

where X_i signifies the position of the i-th grasshopper, S_i denotes the grasshopper's social interaction, G_i stands for the gravity force on the i-th grasshopper, and W_i represents the wind advection. The social interaction is given as follows:

$$S_i = \sum_{\substack{j=1\\j\neq i}}^{M} s\left(e_{ij}\right)\widehat{e_{ij}} \tag{4}$$

In the provided equation, M indicates the count of grasshoppers, s denotes the social force, while e_{ij} represents the Euclidean distance between the i-th and j-th grasshopper. Where $e_{ij} = \left|X_j - X_i\right|$ and $\widehat{e_{ij}} = \frac{X_j-X_i}{e_{ij}}$. The function G_i is defined as follows:

$$G_i = -g\widehat{e_g} \tag{5}$$

where g is the gravitational constant and $\widehat{e_g}$ is the unity vector towards the center of earth. The function W_i is given as:

$$W_i = d\widehat{u_w} \tag{6}$$

In the aforementioned equation, d represents the drift constant, and $\widehat{u_w}$ is the unity vector directed towards the wind. The grasshoppers' position is defined as follows:

$$X_i = \sum_{\substack{j=1\\j\neq i}}^{M} s\left(\left|X_j - X_i\right|\right)\frac{X_j - X_i}{e_{ij}} - g\widehat{e_g} + d\widehat{u_w} \tag{7}$$

The equation is modified to address the optimization problem as illustrated in the following equation:

$$X_i^d = c\left(\sum_{\substack{j=1\\j\neq i}}^{M} c\frac{u_d - l_d}{2}s\left(\left|X_j^d - X_i^d\right|\right)\frac{X_j - X_i}{e_{ij}}\right) + \widehat{O_d} \tag{8}$$

where u_d and l_d are the upper and lower limits of the d-th dimension, respectively, while $\widehat{O_d}$ denotes the optimal solution identified thus far. The variable c signifies as the inertia weight and attraction zone. The value of c is outlined below:

$$\mathrm{c} = \mathrm{c_{max}} - \mathrm{t}\frac{\mathrm{c_{max}} - \mathrm{c_{min}}}{\mathrm{T}} \tag{9}$$

where c_{max} and c_{min} denote the maximum and minimum values respectively, t represents the current iteration, and T signifies the maximum number of iterations.

2) GENETIC ALGORITHM (GA)

GA is a bio-inspired optimization approach that operates based on principles of natural evolution. The GA generates solutions through an optimization process involving genetic operators such as selection, mutation, and crossover. Initially, a random population of chromosomes is generated, with a fitness value calculated for each chromosome [55]. A new population is then formed through the processes of selection, crossover, and mutation. The selection mechanism involves choosing two parent chromosomes. The crossover operation produces new offspring that inherit genetic attributes from these selected parents. Lastly, the mutation mechanism involves introducing minor changes in the new offspring [56]. The subsequent steps of Algorithm 1 outline the GA optimization process:

Algorithm 1 GA

```
Step 1: The initial population of the chromosomes is generated randomly
Step 2: The fitness of each chromosome is evaluated
Step 3: The parents are selected for the mating process.
Step 4: With the crossover and mutation operation new offspring are generated.
Step 5: The new generation replaces the current ones.
Step 6: Go to Step 2 until the termination is completed.
```

3) HYBRID GOA-GA

The proposed hybrid GOA-GA algorithm integrates the unique characteristics of both the GOA and GA algorithms. This approach leverages the exploration capabilities of the GOA and the exploitation capacity of the GA. The GA operation prevents the GOA from falling into local optima [57]. The details of the proposed algorithm are explained as follows.

Step 1: Initialization

The positions of the grasshoppers are randomly allocated in a d-dimensional environment. The fitness function of each agent is calculated according to equation (2). The position of the best initial grasshopper is then selected as the target position.

Step 2: Search process by GOA

The position of each grasshopper is updated in accordance with equation (8). Subsequently, the fitness of each grasshopper candidate is evaluated. The target position, denoted as O, is identified as the most optimal grasshopper position obtained thus far.

Step 3: Selection

In this step, tournament selection (TS) is utilized to choose solutions from the population. With TS, optimal solutions are selected based on a probability assigned to each solution. The chosen candidates from this process are subsequently designated as parents. This TS process is iteratively repeated to ensure the selected candidates are frequently considered.

Crossover:

Following the selection of parents through TS, crossover is executed. An arithmetic crossover or binary crossover is applied to the chromosomes selected with TS. In the case of arithmetic crossover, combination is handled linearly. However, when arithmetic crossover is inapplicable to binary genes, uniform crossover is employed instead.

Algorithm 2 GOA-GA

```
1: Initialize random population of grasshoppers $X_i$ ($i = 1, 2, \ldots\ldots, M$)
2: Initialize $c_{max}$, $c_{min}$, $T$, and set $t = 0$
3: Evaluate the fitness of each search agent using Equation (2)
4: Identify the best search agent and set its position as $O$
WHILE $t < T$ DO
  5: Update c using Equation (9)
  FOR $i = 1$ TO $M$ DO
    6: Update position of the current search agent using Equation (8)
    7: Evaluate its fitness using Equation (2)
    8: If the new solution is better, update $O$
    9: Select parents for crossover from current population using Tournament Selection
    10: Apply crossover and mutation on selected parents
    11: Evaluate the fitness of offspring
    12: If offspring solution is better than the parent, replace the parent with offspring
  END FOR
  13: Update the best global solution if a better solution is found during this iteration
  14: Increment $t$ by 1
END WHILE
15: RETURN $O$ as the optimal solution (features)
```

Mutation:

In this approach, Gaussian mutation is employed where a randomly distributed Gaussian unit is added to the selected gene. The new gene value is clipped if the value of the Gaussian mutation falls outside of the user-specified lower or upper bounds for that gene.

Step 4: Updation

The agents, or candidates obtained thus far, are evaluated using the fitness function. The position of the best agent is then updated to serve as the new target position.

Step 5: Termination

The proposed algorithm is terminated once the maximum number of generations has been reached. The optimal solution is selected from the final target position.

4) PSEUDOCODE FOR THE GOA-GA METHODOLOGY

The following is Algorithm 2, which presents the pseudocode for the proposal:

5) COMPUTATIONAL COMPLEXITY OF OUR FEATURE SELECTION APPROACH

To calculate the computational complexity of our approach, we analyze each step of Algorithm 2 and determine how the computational effort scales with the size of the input. In our case, the key factors are the number of grasshoppers (search agents) M, the number of dimensions (features) d for each grasshopper, and the number of iterations T. Let's break it down:

Computational Complexity of Our Feature Selection Approach (GOA-GA)

1- Initialization (Steps 1-4):

Grasshoppers Initialization: $O(Md)$.This phase accounts for initializing M grasshoppers, each with d dimensions, which represents a linear scaling with the size of the population and the dimensionality of the problem.

Constants Initialization: $O(1)$. The setup of fixed parameters like c_{max}, c_{min}, and T.

Step 3: $O(Mf)$ per iteration, where f is the complexity of the fitness function.

Step 4: $O(M)$. as it involves a linear scan to identify the best agent.

2- Main Loop (Step 5 and WHILE loop):

Update of c: $O(1)$ per iteration. A simple calculation that's independent of M or d.

Iterations Count: The loop runs for T iterations, contributing to the overall complexity.

3- Within Each Iteration (FOR loop Steps 6-12):

Position Update (Step 6): $O(M^2d)$. Each grasshopper's position update involves pairwise comparisons across dimensions, resulting in quadratic complexity due to these comparisons between grasshoppers.

Fitness Evaluation (Step 7): $O(Mf)$ per iteration, where f is the complexity of the fitness function. GA Operations (Steps 9-12): $O(Mg)$ for all grasshoppers per iteration, where g represents the collective complexity of selection, crossover, and mutation.

4- End of Iteration (Steps 13-14):

Global Solution Update: $O(d)$

Incrementing t: $O(1)$ per iteration.

5- Final Step (Step 15):

Returning the Optimal Solution: $O(1)$.

Generally Computational Complexity:

Overall Complexity $= O(T \times (M^2d + Mf + Mg))$

The overall complexity of our approach effectively encapsulates the impact of the main iterative loop, along with the intricacies of fitness evaluation and genetic algorithm operations. This complexity is characterized by quadratic scaling with respect to the number of grasshoppers M, primarily due to the position update step. In our case, this complexity is manageable, owing to the judiciously chosen small value of M, which has produced acceptable results. Additionally, there is linear scaling with the number of dimensions d, the number of iterations T, and the complexities of both the fitness function f and the GA operations g. This linear scaling contributes to a more predictable and manageable increase in computational requirements as the problem size grows, suggesting that the algorithm can adapt to datasets of different sizes and complexities, making it versatile. In our approach, the principal determinants of computational complexity are M, d, and T. Here, d signifies the number of features in the dataset, a known and fixed quantity. We have chosen to set M at 15 and T at 30, selections that have provided satisfactory results in our experiments. This configuration strikes a balance between performance efficiency and computational manageability, thereby ensuring effective processing while keeping the computational demands within feasible limits.

Determining the appropriate values for M and T is intrinsically linked to the nature of the dataset. The most effective method to ascertain these values is through empirical testing. Conducting experiments with varying M and T values allows for the observation of their impact on performance metrics. Critical in this process is finding a balance where the number of agents and iterations are sufficient to enable thorough exploration and exploitation of the search space. However, these values should not be so high as to cause excessive and unjustified computational overhead. Inappropriate values for M and T may increase the risk of converging to local optima, hindering the discovery of more globally optimal solutions. Furthermore, an unsuitable value for M, particularly in high-dimensional spaces, could lead to overlooking potentially superior solutions.

- **Challenges:**

-Computational Burden with Large M:

The quadratic scaling of computational complexity with the number of grasshoppers (M^2d) in the position update step can become a significant burden when M is large. This leads to increased computation time, particularly for very large datasets or an excessively high number of grasshoppers. The efficiency may decline due to the quadratic component of the complexity. Additionally, the scalability of our approach is affected by this quadratic dependency on M. As M increases, the computational load escalates, especially in high-dimensional feature spaces.

- Risk of Premature Convergence or Stagnation:

Like many optimization algorithms, there is a risk of premature convergence or stagnation. This is particularly likely if the balance between exploration and exploitation is not effectively managed.

- **Good Aspects:**

-Efficiency and Scalability:

In the context of our datasets, optimal efficiency is observed in the results. Our approach exhibits manageable scalability, characterized by linear scaling with respect to both d and T. Additionally, it effectively navigates the solution space due to a well-balanced exploration and exploitation mechanism.

-Cloud-Based Systems:

In cloud-based environments, where scalable computational resources are plentiful, our approach can be used effectively even for large datasets. The cloud infrastructure can accommodate the increased computational demands. Parallel processing and distributed computing techniques, applicable in cloud environments, can address the quadratic complexity associated with M. Additionally, certain elements of the algorithm, including the evaluation of fitness across multiple grasshoppers and the execution of GA operations, lend themselves to parallelization. This capability could lead to a significant reduction in real-world execution times.

D. DATA SPLITTING

In this stage, each dataset is divided into two distinct parts at a ratio of 75/25. The larger portion, constituting 75% of the data, is used for training the model, while the remaining 25% is reserved for testing and verifying its performance. This practice of splitting the data into training and testing sets is a fundamental step in model validation. It ensures that the model's effectiveness is evaluated on previously unseen data, providing a more accurate assessment of its real-world performance. This process also aids in tuning the model's parameters and identifying potential overfitting, where a model might excel with training data but perform poorly on new, unseen data. Tables 2, 3, and 4 display the statistical distribution of the training and testing sets for each class within each dataset.

E. ADDRESSING THE IMBALANCE IN TRAINING DATA

Imbalanced datasets are a common challenge in various real-world ML applications, including network intrusion detection. The imbalance can undermine classifiers' efficacy, especially when handling minority threat categories within highly imbalanced datasets. To overcome this challenge, two prevalent techniques are employed: oversampling and undersampling [58].

Oversampling: The oversampling approach increases the representation of minority classes either by replicating existing instances or by generating synthetic samples until all classes have an equal count. The most significant advantages of this strategy include the preservation of majority class information, heightened sensitivity to the minority class, and a more comprehensive training process, as the model is trained on additional examples from the minority class. However, it also poses challenges, including potential noise introduction and increased time and computational resources required due to the enlargement of the dataset [59].

Undersampling: The undersampling method aligns the majority class representation with the minority class by reducing the instances of the majority classes. This strategy is computationally efficient and maintains the information of minority classes, but it risks losing valuable information and diversity from the majority classes.

Trade-offs and Our Proposal: Balancing data can appreciably diminish bias, thereby enhancing classifier performance and evaluation metrics, and allowing the model to generalize better from the training data to unseen data. However, this process can incur trade-offs, such as the risk of underfitting or overfitting. Underfitting may lead to a model that is too simplistic to capture the underlying patterns in the data. Overfitting poses a significant challenge as it results in exceptional performance on the training data but poor generalization to new, unseen data [60]. In our proposal, we amalgamated two methodologies—oversampling and undersampling—to exploit their pros while mitigating their cons. Specifically, we employed the ADASYN method, an advanced version of the widely-used SMOTE technique, to increase the samples for the minority class [61]. Simultaneously, we reduced the samples of the majority classes using the RUS technique. Our goal was to optimize performance without bias and with minimal computational cost, both of which are critical considerations in real-time threat detection within the context of networked systems, especially cloud systems.

For the testing dataset, our balancing strategy diverges from that of the training dataset. Our philosophy aligns with a widely recognized ML axiom: the testing dataset should mirror the real-life data distribution as closely as possible to provide a precise assessment of the system's real-world efficacy. Therefore, we preserved the original proportions of the classes in our testing dataset, mimicking those found in the actual dataset used. Tables 2, 3, and 4 depict the quantity of newly balanced training samples resulting from our combined balancing approach. Algorithm 3 presented below offers a concise pseudocode representation of our proposed balancing approach.

Algorithm 3 Balancing Approach (ADASYN-RUS)

```
1. procedure Balance-Training-Dataset
2.   input: An imbalanced training dataset
3.   output: A balanced training dataset
4.   Initialize balanced_training_dataset as an empty dataset
5.   for each class in the imbalanced training_dataset do
6.     if class is a minority class (Number of training Instances $<$ 1000) then
7.       Apply ADASYN to the class such that the total instances in this class become approximately 3000
8.       Add the oversampled class data to balanced_training_dataset
9.     else if class is a majority class (Number of training Instances $>$ 100,000) then
10.      Apply RUS to the class such that the new number of instances = round(old training instances * 0.8)
11.      Add the undersampled class data to balanced_training_dataset
12.    else
13.      Add the class data to balanced_training_dataset unchanged
14.    end if
15.  end for
16.  return balanced_training_dataset
17. end procedure
```

F. CLASSIFICATION

Classifiers aim to categorize incoming packets as either malicious or benign, a task that can be executed using DL and ML methods. However, due to their complexity and high resource demands, DL models may not be ideal for our scenario. ML models, on the other hand, having ample capability to handle our datasets, appear more fitting. Among various ML models, such as SVM, DT, and others, our experiments indicated that the tree-based model exhibits superior performance, requiring less training and testing time compared to other classifiers. Thus, we employed a random forest (RF) classifier in our study, feeding it with the most beat feature set selected.

Random Forest (RF)

RF is a technique that falls under the category of ensemble learning, a process that combines multiple decision trees (DT) to enhance the overall predictive performance. A DT is a supervised machine learning model used for both classification and regression tasks. The possible decisions/outcomes choices are presented in a structure resembling a tree, where every node represents a feature or attribute, each branch signifies a likely value of a feature, and each leaf indicates a numerical value or class label [62]. The training set is used to build the tree, and the purpose of developing this tree is to construct a classifier that can reliably predict the target variable while increasing information gain and decreasing the nodes' number [63]. To avert overfitting, the DT classifier autonomously selects the most valuable features for tree construction and employs a pruning approach to remove superfluous branches. Common kinds of DT include C4.5, CART, and ID3 [42]. In some instances, a single tree may not provide satisfactory performance; thus, techniques such as RF and XGBoost, both consisting of multiple DTs, are utilized [6]. Therefore, the structure of the RF model makes it fit for addressing large datasets, and it also offers an assessment of the most influential variables in the classification process. Our research revealed that the RF method delivers superior performance. Since this method integrates multiple DTs, it produces a model that is both more robust and more accurate.

G. EVALUATION CRITERIA

In this section, we shed light on the evaluative criteria employed to measure the performance of our suggested model. These criteria are formulated based on the confusion matrix (CM), a two-dimensional grid that delineates actual and predicted classes, as illustrated in Table 5 [62]:

TABLE 5. The confusion matrix.

		Predicted	
		Attack	Normal
Actual	Attack	True Positive (TP)	False Negative (FN)
	Normal	False Positive (FP)	True Negative (TN)

- TP: This refers to an intrusion scenario that has been correctly identified as an attack.
- TN: This denotes a non-intrusive sample that is successfully classified as normal.
- FP: This refers to a normal instance that is incorrectly classified as an attack.
- FN: This term pertains to an attack instance that is erroneously identified as benign.

Accuracy (ACC): This metric denotes the proportion of records that have been properly detected relative to the total number of records, thereby indicating the model's effectiveness.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{10}$$

Recall (R): This concept refers to the proportion of actual attacks that were correctly predicted as such, out of the entirety of actual attack instances. It is also known as sensitivity (S), detection rate (DR), or true positive rate (TPR).

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{11}$$

False Alarm Rate (FAR): Also known as the false positive rate (FPR), this term signifies the proportion of samples incorrectly identified as attacks relative to the total count of actual normal samples.

$$\mathrm{FAR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{12}$$

Precision (P): This measure evaluates the proportion of correctly predicted attacks out of the complete set of instances identified as attacks.

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{13}$$

F1-score (F): This is a metric used to evaluate a system's effectiveness by taking into account both its recall and precision, commonly referred to as the F1 measure.

$$\mathrm{F} = \frac{2}{1/\mathrm{Precision} + 1/\mathrm{Recall}} \tag{14}$$

Missed Alarm Rate (MAR): Also referred to as the false negative rate (FNR), this measure denotes the ratio of instances mistakenly identified as normal in relation to the entire pool of instances.

$$\mathrm{MAR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} \tag{15}$$

ACC, R, and FAR are essential metrics that characterize the capabilities and effectiveness of any IDS. Alongside these, we also consider the receiver operating characteristic (ROC) curve. This curve is generated by comparing the model's false positive rate (FPR) and true positive rate (TPR) [64]. Although the ROC curve is traditionally used in the performance assessment of binary classification models [40], in this context, we apply it to multi-class classification.

IV. EXPERIMENT AND RESULTS DISCUSSION

This section introduces, discusses, and compares the findings of our proposition with those of preceding research efforts. The experimental phase was carried out using Python within the Google Colab Pro environment, which boasts a robust 25 GB of RAM. Notably, during the execution, a suite of libraries, including Sklearn and several others, were employed.

Table 6 lists the parameters utilized in the classifiers for this study. In this research, acceptable results were obtained without adjusting the hyperparameters. However, by using the grid search approach for hyperparameter optimization, there is still room for these models to be improved in the future. In order to find the best possible hyperparameters, this approach thoroughly investigates every potential parameter combination. In contrast, the random search strategy picks values at random and might not always produce the best results.

TABLE 6. Values of parameters used in the classifiers.

Method	Parameter Value	Method	Parameter Value
FLN HL: 1	encoded:onehot Dense=50 Hid_layer: activation='relu' HL: 3 Out_layer:activation='softmax' loss='categorical_crossentropy' optimizer='sgd' batch_size= epochs=100	SVM	kernel='linear' gamma='auto'
LSTM		LR	solver='lbfgs'
AlexNet		DT	random_state=50
DNN		XGB	n_estimators=35 random_state=50
DBN		RF	

Table 7 presents the performance criteria of our proposal for multi-class classification across four scenarios: without feature selection approach, GA, GOA, and the combined GOA-GA. The values provided include accuracy and the weighted averages of recall, precision, and F1-measure, as well as the averages of FAR and MAR. Based on the results in Table 7, we found that our hybrid strategy for feature selection, which combined both GOA-GA techniques, yielded a highly meaningful feature set. The random forest classifier, when provided with this ideal feature set, demonstrated outstanding performance within an acceptable timeframe. However, it's essential to note that the effectiveness of the proposed strategy in real-world scenarios remains uncertain. This is because the evaluation was conducted in a lab setting using specific datasets. Factors such as the structure of the dataset and parameter values could influence the model's performance.

Table 7 shows the efficacy of our hybrid strategy compared to each solo feature selection technique, all using the random forest model. The findings show that in terms of classification ACC, R, FAR, P, F, MAR, and processing time, our proposed method consistently exceeds the standalone approaches. We also note that reducing the number of features and finding the optimal ones not only improves performance but also reduced processing time. The numbers of the features that each technique picked are listed in the ''Selected Features'' column. Tables 21, 22, and 23 provide the precise names of each feature that was chosen.

Due to the complexities of high-dimensional data and the inherent drawbacks of any solitary feature selection technique, a hybrid approach to feature selection has been proposed. Complex patterns are frequently present in high-dimensional datasets, which may make it difficult for a single feature selection approach to recognise them correctly. Each technique has advantages and disadvantages of its own. Our hybrid approach seeks to integrate the advantages of two metaheuristic algorithms: GA and GOA. By doing so, it successfully circumvents the constraints intrinsic to each. This strategy makes use of the GA's potential for exploitation and the GOA's exploratory capabilities. The GA operation keeps the GOA from getting trapped in a local optimum scenario. We have found that employing this hybrid method enhances the system's overall performance by providing an ideal subset of the original attributes. In which, Tables 10, 11, and 12 further underscore the robustness of our hybrid approach, showcasing its superiority over prevailing methodologies.

The performance of many ML and DL algorithms, including SVM, LR, FLN, LSTM, AlexNet, DNN, DBN, DT, XGBoost, and RF, is compared in Table 8. This comparison takes into account important assessment criteria, including ACC, R, FAR, and MAR, along with both training and testing times. Please note that Table 6 contains detailed information about the parameters employed by these classifiers. Using the DateTime library, we were able to determine the timepoints for the beginning and finish of the training and testing stages.

Table 8 demonstrates the improved performance of the RF model compared to other models. Upon investigation, we discovered that tree-based models outperform neural network (NN) models and other types in metrics such as ACC, R, FAR, and MAR. Moreover, tree models have the advantage of shorter processing times. It's noteworthy that, although NN models required considerably more processing time than other models, they still surpassed SVM and LR in various aspects. One avenue for enhancing the performance of neural network classifiers, such as FNN, LSTM, AlexNet, DNN, and DBN is to use optimal features derived from the GOA-GA algorithm. This can help fine-tune and optimize the weights of these classifiers. To further optimize the performance of LSTM, AlexNet, DNN, and DBN adjustments to their architecture can be made, including modifying the number of hidden layers, the density of connections, the choice of activation functions, and the number of epochs, among others.

The accuracy and loss of our proposed model during both the training and validation phases are illustrated in Figures 2, 4, and 6, which represent the performance without utilizing our feature selection approach. Conversely, Figures 3, 5, and 7 demonstrate the results following the application of our feature selection methodology (GOA-GA). These metrics are influenced by the number of trees used to form the random forest model. Conversely, for neural network classifiers, such values are often defined by the number of epochs.

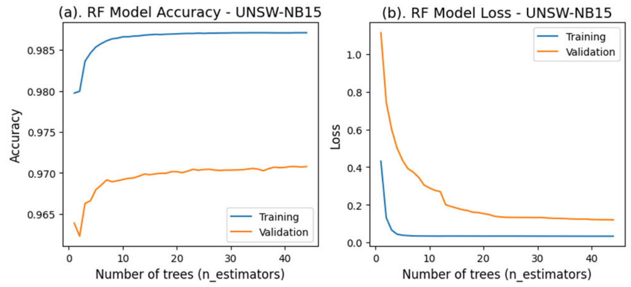


FIGURE 2. Accuracy and loss metrics - UNSW-NB15 (48 Features).

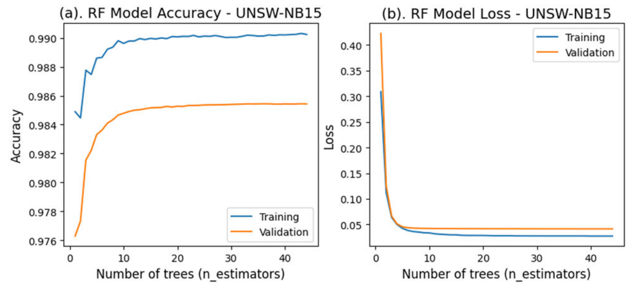


FIGURE 3. Accuracy and loss metrics - UNSW-NB15 (25 Features).

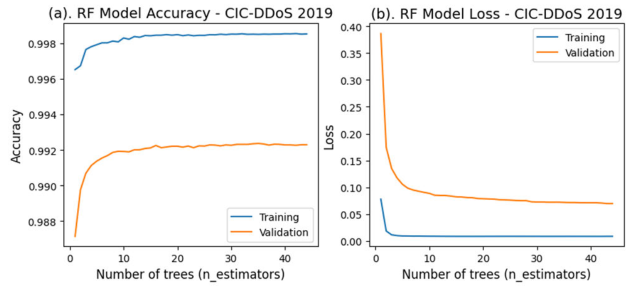


FIGURE 4. Accuracy and loss metrics - CIC-DDoS 2019 (87 Features).

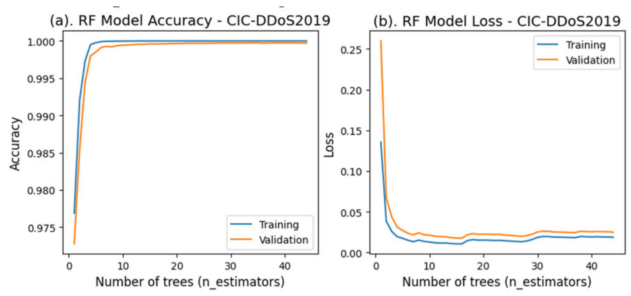


FIGURE 5. Accuracy and loss metrics - CIC-DDoS 2019 (40 Features).

Figure 2 depicts the model achieving its highest accuracy of 97.05% and its lowest loss of 0.127% at the 35th tree.

TABLE 7. Findings for each selection method and our suggested hybrid approach.

Dataset	Feature Selection	F. No	Selected Features	ACC	R	FAR	P	F	MAR	Train T (s)	Test T (s)
UNSW-NB15	Without	48	All features except No 47	97.05	97.05	0.358	97.04	97.06	13.558	48.828	1.907
	GA	26	2, 3, 7, 9, 11, 14, 16, 17, 18, 21, 22, 23, 24, 25, 27, 32, 33, 34, 38, 39, 42, 43, 44, 45, 46, 48	97.98	97.98	0.143	97.94	97.95	11.374	16.869	1.328
	GOA	25	2, 3, 6, 7, 8, 9, 10, 11, 15, 16, 17, 18, 21, 23, 24, 25, 27, 30, 31, 38, 41, 42, 44, 46, 48	98.11	98.11	0.098	98.22	98.12	8.207	15.892	0.985
	GOA-GA	**25**	**0, 1, 2, 3, 9, 10, 11, 13, 15, 17, 18, 19, 22, 23, 24, 25, 26, 27, 29, 35, 37, 38, 39, 41, 48**	**98.54**	**98.54**	**0.040**	**98.51**	**98.52**	**7.854**	**16.304**	**1.063**
CIC-DDoS 2019	Without	87	All features except No 87	99.23	99.23	0.006	99.28	99.30	0.962	53.941	2.192
	GA	37	2, 3, 5, 6, 7, 9, 11, 12, 13, 14, 18, 19, 20, 22, 23, 28, 30, 31, 36, 39, 41, 42, 47, 49, 50, 51, 52, 57, 59, 61, 65, 66, 71, 72, 78, 79, 80	99.91	99.91	0.003	99.87	99.89	0.098	30.784	1.573
	GOA	50	0, 2, 3, 6, 7, 8, 10, 11, 12, 14, 16, 20, 21, 23, 24, 26, 29, 30, 31, 33, 37, 39, 40, 41, 42, 44, 45, 46, 47, 48, 50, 51, 52, 54, 58, 59, 61, 63, 67, 68, 69, 70, 72, 73, 77, 80, 81, 82, 83, 86	99.85	99.85	0.003	99.80	99.81	0.487	42.118	1.696
	GOA-GA	**40**	**0, 1, 3, 4, 5, 6, 7, 8, 9, 10, 14, 15, 16, 18, 20, 21, 25, 26, 28, 33, 35, 37, 42, 43, 44, 45, 46, 49, 53, 55, 61, 62, 63, 66, 69, 71, 73, 75, 76, 83**	**99.97**	**99.97**	**0.002**	**99.97**	**99.98**	**0.023**	**34.383**	**1.594**
CIC Bell DNS EXF 2021	Without	42	All features except No 27	88.52	88.52	6.995	88.68	88.32	17.294	45.651	1.062
	GA	19	0, 1, 13, 15, 24, 25, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41	91.17	91.17	3.839	91.24	91.28	15.120	36.179	0.259
	GOA	24	0, 1, 6, 13, 14, 15, 16, 19, 20, 24, 25, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41	90.76	90.76	4.779	90.87	90.93	16.029	39.684	0.778
	GOA-GA	**17**	**0, 1, 13, 24, 28, 29, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41**	**92.28**	**92.28**	**3.357**	**92.78**	**92.06**	**14.065**	**34.436**	**0.018**

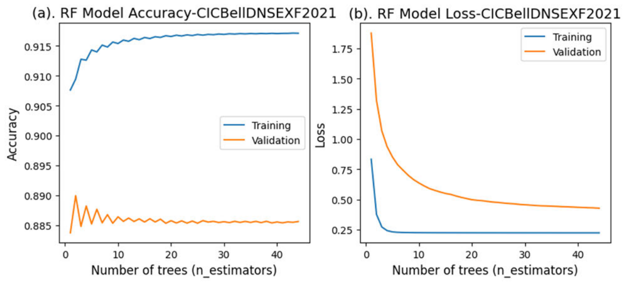


FIGURE 6. Accuracy and loss metrics - CIC Bell DNS EXF 2021 (42 Features).

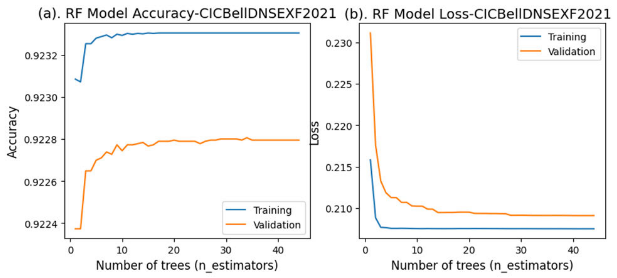


FIGURE 7. Accuracy and loss metrics - CIC Bell DNS EXF 2021 (17 Features).

The training accuracy curve, as elaborated in Figure 2a, initially ascends from 98.00% to 98.71% before stabilizing. Concurrently, the validation accuracy starts at 96.42% and reaches its peak of 97.05% at the 35th tree. This results in a disparity of 1.66% between the peak training and validation accuracies. Figure 2b illustrates that the training loss commences at 0.43% and gradually decreases to 0.033% before achieving stability. In contrast, the validation loss starts at 1.12% and diminishes to its lowest value of 0.127% by the 35th tree, at which point it also stabilizes. Consequently, this leads to a 0.094% discrepancy between the lowest training and validation losses.

Meanwhile, as depicted in Figure 3, the model's accuracy peaks at 98.54% with its smallest loss at 0.041% in the 35th tree. The training accuracy curve is detailed further in Figure 3a, where it rises from 98.49% to 99.01% before stabilizing. Meantime, the validation accuracy begins at 97.63% and reaches its maximum of 98.54% with the 35th tree. Consequently, there is a 0.47% variation between the maxima of training and validation accuracies.

In Figure 3b, the training loss begins at 0.32% and eventually drops to 0.027% before stabilizing. The validation loss, in contrast, starts off at 0.43% and drops to its lowest value of 0.041% by the 35th tree, in which it also stabilizes. As a result, there is a 0.014% discrepancy between the lowest training and validation losses.

In a similar vein, Figure 4 illustrates the model achieving its maximum accuracy of 99.23% and its minimal loss of 0.072% at the 35th tree. Figure 4a presents an overview of the accuracy trends. Here, the training accuracy initiates at 99.64%, gradually increases to 99.85%, and subsequently stabilizes. In parallel, the validation accuracy commences at

TABLE 8. Comparison of our proposed solution using RF and other classifiers.

Modle	ACC	R	FAR	MAR	Trian Time/ s	Test Time/ s
UNSW-NB15						
SVM	96.63	96.63	1.057	15.92	3780.81	207.582
LR	96.94	96.94	0.873	11.33	146.015	0.113
FLN	97.32	97.32	0.814	8.369	866.526	41.561
LSTM	98.46	98.46	0.068	7.159	4108.17	36.316
AlexNet	98.13	98.13	0.498	7.965	1620.88	22.903
DNN	98.05	98.05	0.211	23.43	1042.95	34.677
DBN	98.11	98.11	0.204	19.62	1303.86	31.311
DT	97.76	97.76	0.431	9.925	4.968	0.521
XGBoost	98.24	98.24	0.352	8.979	247.446	1.159
RF	98.54	98.54	0.040	7.854	16.304	1.063
CIC-DDoS 2019						
SVM	98.70	98.70	0.109	1.173	827.028	581.465
LR	94.62	94.62	0.454	4.702	393.086	0.354
FLN	98.43	98.43	0.133	3.671	802.676	29.136
LSTM	99.64	99.64	0.002	0.021	4466.35	57.07
AlexNet	98.57	98.57	0.152	1.627	1704.65	13.558
DNN	98.79	98.79	0.134	3.664	1404.71	23.474
DBN	98.32	98.32	0.184	4.291	1821.37	59.384
DT	98.99	98.99	0.004	0.086	14.607	1.074
XGBoost	99.92	99.92	0.001	0.018	399.623	1.149
RF	99.97	99.97	0.002	0.023	34.383	1.594
CIC Bell DNS EXF 2021						
SVM	84.65	84.65	7.392	21.68	9143.65	896.795
LR	78.97	78.97	8.654	24.00	183.164	0.050
FLN	87.78	87.78	6.130	18.89	982.771	8.089
LSTM	90.88	90.88	5.434	17.62	4467.90	37.279
AlexNet	87.15	87.15	5.570	18.50	1901.76	10.47
DNN	86.43	86.43	4.892	15.77	1524.80	36.702
DBN	88.95	88.95	4.278	14.89	2319.94	57.155
DT	92.26	92.26	3.376	14.08	6.980	0.102
XGBoost	92.27	92.27	3.361	14.07	98.085	0.037
RF	92.28	92.28	3.357	14.06	34.436	0.018

98.75% and peaks at 99.23% at the 35th tree, resulting in a 0.62% disparity between the highest points of training and validation accuracies. Figure 4b, conversely, demonstrates the training loss divergence pattern, beginning at 0.078% and progressively declining to 0.009% before reaching stability. Similarly, the validation loss starts at 0.386% and diminishes to its lowest value of 0.072% at the 35th tree, where it also stabilizes. Consequently, there is a 0.063 difference between the lowest points of training and validation losses.

On the other hand, Figure 5 depicts the model's 35th tree as the point at which it reaches its maximum accuracy of 99.97% and its minimal loss of 0.024%. An overview of the accuracy measures is shown in Figure 5a. The training accuracy starts at 97.76%, increases gradually to 100%, and then stabilizes. Parallel to this, the validation accuracy begins at 97.18% and peaks at 99.97% at the 35th tree. As a consequence, there is a 0.03% difference between the highest points of training and validation accuracy.Figure 5b, on the other hand, illustrates the training loss divergence pattern. It starts at 0.14% and progressively decreases to 0.019%, where it stabilizes. Therefore, the validity loss starts at 0.26% and drops till it gets to its smallest value at the 35th tree, when it likewise stabilises. As a result, there is a 0.005 difference between the lowest values of the training and validation losses.

Similarly, Figure 6 reveals that at the model's 35th tree, it attains its highest accuracy of 88.52% and its minimal loss of 0.453%. Figure 6a provides a summary of the accuracy trends. Here, the training accuracy initiates at 90.57%, progressively advances to 91.71%, and eventually reaches equilibrium. In tandem, the validation accuracy commences at 88.46% and culminates at 88.52% at the 35th tree. Consequently, this leads to a 3.19% variance between the peak values of training and validation accuracy. Conversely, Figure 6b displays the pattern of training loss reduction. This pattern begins at 0.84% and methodically diminishes to 0.223%, at which point it becomes steady. In a similar manner, the validation loss starts at 1.87% and declines to its lowest mark of 0.453 at the 35th tree, where it also stabilizes. Thus, there is a 0.23 discrepancy between the minimal values of the training and validation losses.

Correspondingly, Figure 7 demonstrates that at the model's 35th tree, it achieves its highest accuracy of 92.28% and its lowest loss of 0.209%. Figures 7a and 7b display the accuracy and loss metrics, respectively. The methodology for their interpretation follows the same approach as that applied to the previously discussed figures. The discrepancies between the training and validation accuracies at the 35th tree are 0.05, and for the losses, they are 0.0016.

In summary, Tree No. 35 produced acceptable outcomes in terms of both training and testing timeframes, as well as accuracy and loss rates, as illustrated in Table 8 and Figures 3, 5, and 7. Although we could introduce more trees, the potential increase in accuracy and decrease in loss rates might be slight and not worth the extra computing effort.

It's crucial to remember that when a technique is underfitting, both the training and validation performances are subpar, and the losses are significant. This indicates that the classifier hasn't adequately grasped the training data set. In contrast, overfitting results in very high training and significantly lower validation accuracies, coupled with a very low training loss and much higher validation loss. This translates to a large gap in the curve between training and validation. Such a discrepancy is a sign that the classifier has memorized the training data too well but is unable to perform well on unseen data. The training and validation plots of a well-trained classifier should demonstrate high accuracy and low loss for both the training and validation sets. This means that the classifier can produce precise predictions for new data. However, if there's a disparity between the two curves, it suggests a performance difference between the training and validation datasets.

Table 9 presents the variance between the peak values of training and validation accuracy across the datasets, referred to as 'model accuracy', and the difference between the minimal points of training and validation loss, denoted as 'model loss'. Additionally, it includes the accuracy and loss values of validation. These values are extrapolated from the data depicted in Figures 2-7, specifically at Tree No. 35.

TABLE 9. Accuracy and loss metrics across the datasets before and after applying feature selection.

Feature Selection	ACC	Loss	Model ACC	Model Loss	Train Time (s)	Test Time (s)
UNSW-NB15						
Without	97.05	0.127	1.66	0.094	48.828	1.907
GOA-GA	98.54	0.041	0.47	0.014	16.304	1.063
CIC-DDoS 2019						
Without	99.23	0.072	0.62	0.063	53.941	2.192
GOA-GA	99.97	0.024	0.03	0.005	34.383	1.594
CIC Bell DNS EXF 2021						
Without	88.52	0.453	3.19	0.23	45.651	1.062
GOA-GA	92.28	0.209	0.05	0.0016	34.436	0.018

Table 9 illustrates that the feature selection methodology not only reduces processing time and enhances accuracy, along with other evaluation metrics (as indicated in Table 8), but also minimizes overfitting. This is evidenced by the decreased variance between the training and validation accuracies, as well as between the training and validation losses. Thus, our approach has demonstrated robust training and the capability to make precise predictions on new data. It showed superior performance on the CIC-DDoS 2019 dataset compared to the other datasets.

Nevertheless, as evident from Figures 3, 5, and 7, and in consideration of the information previously provided, our model shows very little to negligible overfitting, especially with the UNSW-NB15 dataset. Future work could address this issue by either employing a more effective feature selection method or using a different classifier with optimized hyperparameter tuning.

In Tables 10, 11, and 12, we evaluate the performance of our suggestion in comparison with previously published works. These tables demonstrate that the proposed model surpasses other methodologies, especially in terms of ACC, R, and FAR. This underscores the success of our feature selection methodology. Adopting feature selection approaches has significantly enhanced performance metrics. We used three benchmark public datasets, which encompass the most prevalent current threats, to evaluate our approach. Although some experiments in the literature utilize more advanced methods, it's essential to consider both interpretability and complexity when choosing techniques for classification. In some cases, simpler strategies may exceed more complicated ones. It's important to recognize that performance is influenced by a variety of factors. These include the dataset's size, its preparation, the approach to dimensionality reduction, the balance of the training set, available computing power, classifier parameter settings, and more. It's not just the complexity of the classifier or methodology that matters. The recommended methodology has yielded improved results, primarily due to the efficiency of the feature selection section. This stage is essential for developing any successful IDS. Moreover, robust dataset preparation and addressing the imbalanced dataset issue are crucial. The use of a potent classifier, such as the RF, which has proven effective in different scenarios, further enhances the system's performance.

Table 10 indicates that, while our methodology outperforms most others, there are notable exceptions. Research No. 3 excels in terms of R and F metrics. Proposal No. 8 surpasses in terms of the R and FAR measures. Studies No. 30 and 37 stand out for their performance in P. Approach No. 32 is superior across all metrics. Work No. 35 outperforms in nearly all criteria except for FAR; notably, it utilized a deep learning classifier that required "27,123.87" seconds for processing. Study No. 36 shines in both ACC and FAR metrics, and methodology No. 39 is particularly strong in terms of ACC.

Table 11 demonstrates that although our technique outstands others most of the time, there are some notable exceptions. In terms of the R and P metrics, Research No. 6 stands out. Excluding its performance on the FAR, Approach No. 10 excels across all parameters. Meanwhile, Methodology No. 14 is especially strong in terms of ACC.

Table 12 displays satisfactory performance. It's worth noting that, unlike our work which incorporates both stateless and stateful features, other studies do not. Additionally, very few published studies test their methodologies using this dataset.

We want to be clear that while we are not arguing that our findings are the most impressive ever obtained, our study does show enhanced performance in comparison to the publications referenced in our work for the majority of the metrics. We would like to emphasize that, while the related works and the approaches described in Tables 10, 11, and 12 present novel methodologies, they do not introduce new or innovative techniques. Nonetheless, these studies have contributed to the growing body of knowledge in the IDS sector. Additionally, several of the research papers mentioned used the RF model, with varying degrees of effectiveness. However, the disparities in performance can be attributed to the diverse dataset preparation techniques, feature selection methods, and so on. In contrast, our study employs a standard RF classifier and integrates contemporary strategies for feature selection. Still, we believe that the uniqueness of our research exists in the meticulous combination of both GA and GOA approaches to select the best attributes after the datasets have been adequately processed and balanced. This clarification should underscore the significance and applicability of our approach.

The abbreviations found in Tables 10 and 11 are elaborated in Table 20 with their full names. These full names are not listed in the 'Related Works' section.

Figures 8, 9, and 10 display the confusion matrices for our experiment. The confusion matrix is a commonly utilized machine learning technique that thoroughly evaluates a model's performance based on a dataset. It also aids in determining the classifier's advantages and disadvantages. For instance, the matrix can highlight the categories in which the classifier excels and those where it encounters challenges.

Based on Figure 8, the assessment criteria for every category in the UNSW-NB15 dataset are generated and presented in Table 13. The results highlight robust performance,

TABLE 10. Comparative analysis of the proposed methodology and contemporary techniques using the UNSW-NB15 dataset.

No	Ref. No. and Year	Feature Selection	No. of Features	Methods	ACC	R	FAR	P	F
1	[65] 2016	-	-	DT	85.56	-	15.78	-	-
2	[42] 2017	GA-LR	20	DT	81.42	-	6.39	-	-
3	[66] 2017	-	-	Ramp-KSVCR	93.52	98.68	2.46	-	98.72
4	[67] 2017	PCA	10	GMM	96.70	95.60	3.5	-	-
5	[68] 2017	Weka-ML	5	RF	82.99	83.00	0.061	81.20	81.4
6	[69] 2017	CS- RFE	16	RF	95.09	-	2.415	-	-
7	[70] 2017	PCA	15	GAA-ADS	92.8	91.30	5.1	-	-
8	[71] 2018	-	-	PSO-ANN	91.87	98.61	0.0186	-	-
9	[72] 2018	DBN (FE)	-	Ensemble SVM	-	97.21	-	90.47	93.72
10	[73] 2018	CFS	6	ABC-AFS	95.00	88.00	2.1	-	-
11	[74] 2018	FI	11	RF	75.66	76.00	-	75.00	73.0
12	[75] 2018	-	-	Parallel K-medoids + KNN	94.00	91.6	6.5	93.4	-
13	[76] 2018	IG	-	2-stage classifier	85.78	-	15.64	-	-
14	[33] 2019	BBA (FSFF+CAFF)	26	RF	97.09	95.53	2.03	-	-
15	[40] 2019	-	-	2-layers DNN	66.00	66.00	-	62.30	59.60
16	[77] 2019	A-PCA	-	I-ELM	70.51	77.36	35.09	-	-
17	[78] 2019	TSDL(D-SAE)	10	Softmax	89.13	-	0.7495	-	-
18	[79] 2019	-	-	ICVAE-DNN	89.08	95.68	19.01	86.05	90.61
19	[80] 2019	PSO-ACO-GA	19	2-stage ensemble	91.27	91.30	8.90	91.60	-
20	[41] 2020	PSO-GWO-FFA-GA	30	J48	90.48	97.14	14.95	84.14	90.17
21	[60] 2020	-	-	OSS-SMOTE + CNN-BiLSTM	77.16	79.91	-	82.63	81.25
22	[63] 2020	IG	13	DT	84.83	-	2.01	-	-
23	[81] 2020	NSGA2-MLR	11	NBTree	66.00	64.90	3.85	-	-
24	[82] 2020	-	20	MSCNN-LSTM	95.60	-	9.8	-	-
25	[83] 2020	WFEU	22	FFDNN	77.16	-	-	-	-
26	[84] 2020	FE	-	DBN	85.73	-	-	-	-
27	[85] 2020	XGBoost	19	ANN	77.51	77.53	-	79.50	77.28
28	[86] 2020	CFS	33	ANN	96.44	50.40	-	-	-
29	[11] 2021	-	-	CAE+OCSVM	94.28	96.49	5.51	-	95.06
30	[17] 2021	-	-	OCNN(LSO)–HMLSTM	96.33	95.87	5.87	100	98.13
31	[87] 2021	TS	16	RF	83.12	-	3.7	-	-
32	[88] 2021	Fisher Score, MI, and Spearman Correlation	9	Stacking (Decision Jungle as meta-learner)	99.8	99.9	0.004	99.8	99.8
33	[89] 2021	-	-	MFFSEM	88.85	80.44	2.27	93.88	86.64
34	[6] 2022	Selectkbest	20	Stacking (DT-RF-XGBoost)	94.00	94.00	0.06	-	-
35	[9] 2022	ABC	36	BWO-CONV-LSTM	98.67	98.78	0.045	100	98.77
36	[22] 2022	SOA	-	EEDTL	99.91	96.06	0.008	94.93	-
37	[34] 2023	IG-CS-PSO	21	RF	98.39	98.39	0.046	98.54	98.46
38	[35] 2023	ISSA	14	SMOTE–Tomek + RF	85.15	85.15	-	85.79	85.31
39	[90] 2023	LASSO+ MI+DT+LGB	-	Stacked-Ensemble (AB+LR+KNN) + XGB	99.32	75.79	-	89.94	82.26
Our Proposal		**GOA-GA**	**25**	**RF**	**98.54**	**98.54**	**0.040**	**98.51**	**98.52**

particularly in terms of ACC, R, FAR, and MAR, demonstrating the efficacy of our methodology. Among all categories, the "Worms", "Shellcode", "normal", and "generic" types display the best findings across all measures.

Figure 11 and Tables 14 and 15 illustrate that, for the various classes in the UNSW-NB15 dataset, our method surpassed traditional methods in terms of ACC, R, and FAR. The majority of threats were accurately determined, with an increase in R and a decrease in FAR.

Figure 11 compares the effectiveness of our proposal with other studies in terms of ACC across all categories in the UNSW-NB15 dataset. Our method outperformed the others in all categories, with the exception of the "fuzzers" category from the first study and the "normal" class from the second study.

Table 14 demonstrates the success of our proposal in identifying several types of threats in the UNSW-NB15 dataset. However, there were exceptions: the "dos" and "exploits" types in study No. [66], the "worms" category in article No. [76], the "normal" class in works No. [34] and [86], and the "reconnaissance" and "analysis" classes in reference No. [34].

Table 15 shows that our approach has an acceptable FAR. Although the three references [34], [40], and [68] outperformed our approach in specific classes, overall, our approach performed better on other metrics.

According to Figure 9, we have constructed Table 16 which presents the assessment norms for each type in the CIC-DDoS 2019 dataset. From this, we observed superior findings.

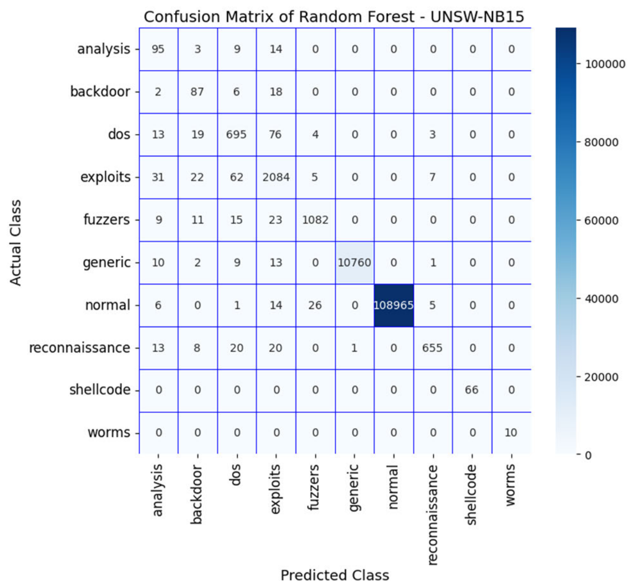
Table 17 compares the DR of our technique with those of previously published methods. Our proposed method surpassed the performance in all categories of the CIC-DDoS 2019 dataset, with the exception of the "Syn" class in reference No. [88].

TABLE 11. Comparative analysis of the suggested approach and state-of-the-art methodologies using the CIC-DDoS 2019 dataset.

No	Ref. No. and Year	Feature Selection	No. of Features	Methods	ACC	R	FAR	P	F
1	[91] 2020	-	87	CNN	95.4	92.4	-	93.3	92.8
2	[92] 2020	-	-	LSTM-Fuzzy	-	93.13	0.022	97.89	-
3	[93] 2020	-	-	HOSVD (Gradient Boosting)	99.87	99.86	0.002	-	-
4	[31] 2021	-	-	GRU	99.7	99.79	-	99.83	-
5	[32] 2021	AE	24	MLP	98.34	98.48	-	97.91	98.18
6	[44] 2021	-	69	DNN	99.97	99.98	-	99.99	99.98
7	[88] 2021	Fisher Score, MI, and Spearman Correlation	13	Stacking (Decision Jungle as meta-learner)	97	97	0.067	99	97.8
8	[94] 2021	SNN	82	MLP	-	79.41	-	91.16	79.39
9	[95] 2022	IG	40	CNN	99.30	-	-	-	-
10	[96] 2022	RFECV + SHAP	5	ADASYN + KNORA-U	99.99	99.99	-	99.99	99.99
11	[97] 2022	-	-	Resilient Backpropagation Neural Network	97.07	99.94	0.058	94.47	97.13
12	[98] 2022	-	8	LSTM	99.83	-	-	-	-
13	[99] 2022	BHO (BSO+ HGSO)	20	WMV ensemble (6 classifiers)	99.41	99.34	0.498	-	99.47
14	[90] 2023	LASSO+ MI+DT+LGB	-	Stacked-Ensemble (AB+LR+KNN) + XGB	99.98	99.83	-	99.94	99.77
Our Proposal		**GOA-GA**	**40**	**RF**	**99.97**	**99.97**	**0.002**	**99.97**	**99.98**

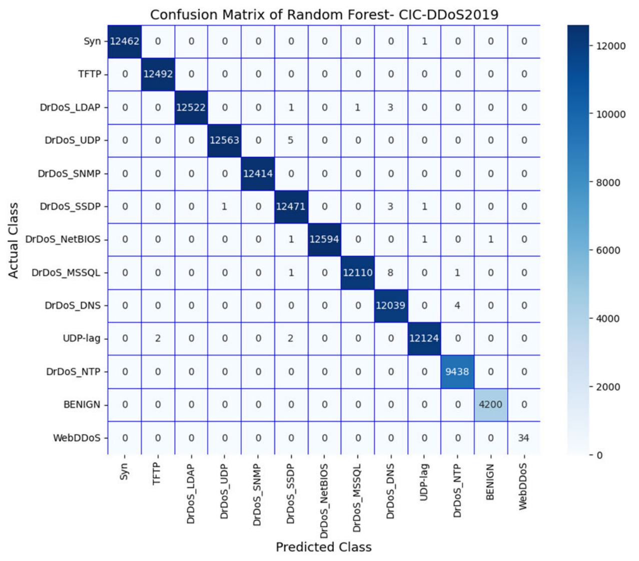
TABLE 12. Comparative analysis of the proposed method and recent studies using the CIC Bell DNS EXF 2021 dataset.

No	Ref. No. and Year	Feature Selection	No. of Features	Methods	ACC	R	FAR	P	F
1	[50] 2021	-	-	MLP	95.32	-	-	91.13	93.12
2	[48] 2022	Extracted Features	-	SVM	91.0	89.7	-	92.3	91.0
3	[100] 2023	-	-	Ensemble of One Rule	-	94.00	-	80.00	-
Our Proposal		**GOA-GA**	**17**	**RF**	**92.28**	**92.28**	**3.357**	**92.78**	**92.06**



	analysis	backdoor	dos	exploits	fuzzers	generic	normal	reconnaissance	shellcode	worms
analysis	95	3	9	14	0	0	0	0	0	0
backdoor	2	87	6	18	0	0	0	0	0	0
dos	13	19	695	76	4	0	0	3	0	0
exploits	31	22	62	2084	5	0	0	7	0	0
fuzzers	9	11	15	23	1082	0	0	0	0	0
generic	10	2	9	13	0	10760	0	1	0	0
normal	6	0	1	14	26	0	108965	5	0	0
reconnaissance	13	8	20	20	0	1	0	655	0	0
shellcode	0	0	0	0	0	0	0	0	66	0
worms	0	0	0	0	0	0	0	0	0	10

FIGURE 8. The confusion matrix for the UNSW-NB15 dataset.



	Syn	TFTP	DrDoS_LDAP	DrDoS_UDP	DrDoS_SNMP	DrDoS_SSDP	DrDoS_NetBIOS	DrDoS_MSSQL	DrDoS_DNS	UDP-lag	DrDoS_NTP	BENIGN	WebDDoS
Syn	12462	0	0	0	0	0	0	0	0	1	0	0	0
TFTP	0	12492	0	0	0	0	0	0	0	0	0	0	0
DrDoS_LDAP	0	0	12522	0	0	1	0	1	3	0	0	0	0
DrDoS_UDP	0	0	0	12563	0	5	0	0	0	0	0	0	0
DrDoS_SNMP	0	0	0	0	12414	0	0	0	0	0	0	0	0
DrDoS_SSDP	0	0	0	1	0	12471	0	0	3	1	0	0	0
DrDoS_NetBIOS	0	0	0	0	0	1	12594	0	0	1	0	1	0
DrDoS_MSSQL	0	0	0	0	0	1	0	12110	8	0	1	0	0
DrDoS_DNS	0	0	0	0	0	0	0	0	12039	0	4	0	0
UDP-lag	0	2	0	0	0	2	0	0	0	12124	0	0	0
DrDoS_NTP	0	0	0	0	0	0	0	0	0	0	9438	0	0
BENIGN	0	0	0	0	0	0	0	0	0	0	0	4200	0
WebDDoS	0	0	0	0	0	0	0	0	0	0	0	0	34

FIGURE 9. The confusion matrix for the CIC-DDoS 2019 dataset.

Table 18 depicts that our strategy has a satisfactory FAR. However, the method described in reference [95] outperformed our approach in specific types. Nevertheless, our system performed better on the ACC metric across all categories.

We have constructed Table 19 based on Figure 10, which details the assessment norms for each category in the CIC Bell DNS EXF 2021 dataset. The results obtained from this were satisfactory.

To the best of our understanding, we couldn't identify any prior research that conducted a comparable analysis of performance metrics for the CIC Bell DNS EXF 2021 dataset to juxtapose against our findings.

The ROC curves for the multi-class classification are shown in Figures 12, 13, and 14. They were drawn depending

TABLE 13. Evaluation metrics across all categories in the UNSW-NB15 dataset.

Category	ACC	R	FAR	P	F	MAR
Normal	99.96	99.95	0.000	100	99.98	0.05
DoS	99.81	85.80	0.098	85.07	85.43	14.20
Fuzzers	99.92	94.91	0.028	96.87	95.88	5.09
Exploits	99.76	94.25	0.145	92.13	93.18	5.74
Generic	99.97	99.67	0.001	99.99	99.83	0.32
Reconnaissance	99.94	91.35	0.013	97.61	94.38	8.65
Backdoor	99.93	76.99	0.052	57.24	65.66	23.01
Analysis	99.91	78.51	0.067	53.07	63.33	21.49
Worms	100	100	0.000	100	100	0.00
Shellcode	100	100	0.000	100	100	0.00

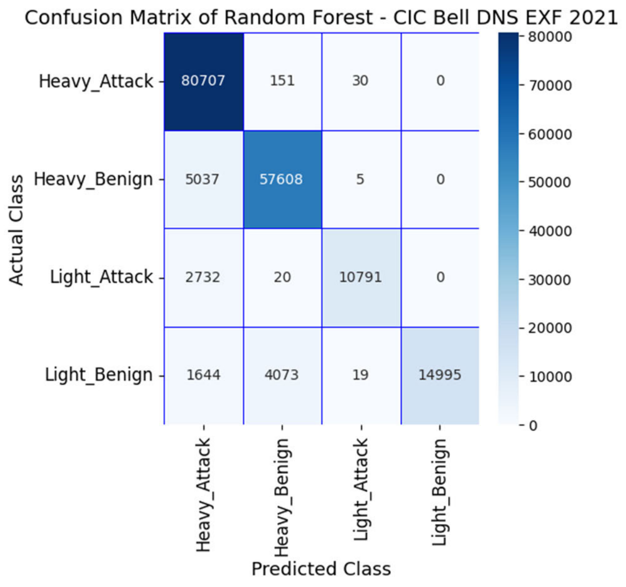


FIGURE 10. The confusion matrix for the CIC Bell DNS EXF 2021 dataset.

on the FPR and TPR values for every category presented in Tables 13, 16, and 19.

The ROC curve offers insightful information about how well the strategy performs for every single class. For example, the proposition had the worst performance in the "backdoor" type in the UNSW-NB15 dataset. On the other hand, in the CIC-DDoS 2019 dataset, the "MSSQL" category showed the poorest performance. Meanwhile, in the CIC Bell DNS EXF 2021 dataset, the "Light-Benign" type demonstrated the weakest performance. A macro-averaged ROC curve also allows us to see how well the model performed across different categories.

As a result, the suggested methodology demonstrated increased efficiency and a shorter processing time—a crucial consideration for cloud systems that prefer to conserve computational resources. This improvement was achieved through meticulous dataset preparation, addressing balance issues, and selecting an ideal set of features using our feature selection strategy. Furthermore, the superior performance of the RF model was notable. The system we designed can effectively detect breaches in future cloud settings, boasting a high Recall and a low FAR. Our findings show that our approach can recognize possible cloud threat patterns, such as those listed in the datasets used. Therefore, it is suitable for operation in cloud environments.

V. CHALLENGES AND BOUNDARIES OF THE APPROACH

1. Data Preprocessing Steps: In feature encoding, utilizing label encoding for nominal features may inadvertently introduce bias, particularly when dealing with categorical data that possesses an underlying ordinal structure. This can create a false sense of order or importance among different categories. For instance, a model might erroneously interpret "category 2" as being more superior or significant than "category 1". Thus, label encoding may not be suitable for all categorical variables. In such contexts, one-hot encoding often emerges as a more appropriate alternative. However, it is important to note that in our specific datasets, the majority of the features are numerical; therefore, we have opted to use label encoding. Additionally, in feature scaling, the choice of a scaling method can profoundly influence the final performance of a model. Min-Max scaling, in particular, is susceptible to the presence of outliers. When a dataset contains pronounced outliers, this technique can distort the relative distances between feature values, leading to skewed interpretations. However, the implementation of a random forest model may mitigate the effect of any outliers. Nonetheless, we have observed satisfactory results with the application of Min-Max scaling in our case.
2. Hybrid Feature Selection Approach: The hybrid approach of utilizing both the GOA and GA presents certain challenges, particularly with regard to computational expense and time consumption, especially when dealing with high-dimensional data. Although the incorporation of GA aims to counteract GOA's tendency to converge to local optima, this hybrid method may still be vulnerable to becoming entrapped in these suboptimal solutions. Furthermore, there is a potential risk of overfitting, a condition in which the selected features may not adequately generalize to unseen data or different datasets. Nevertheless, these challenges can be substantially mitigated through meticulous parameter tuning and by determining an appropriate number of generations. Additionally, our utilization of a hybrid balance approach (ADASYN-RUS) assists in reducing data dimensionality somewhat. Proper implementation of these techniques has led to convergence towards robust performance in our results. Consequently, this affirms the potential effectiveness of our hybrid approach within the context of our selected datasets, demonstrating advantages over standalone GOA or GA methods or other existing methodologies.
3. Imbalanced Data Approach: Over-sampling the minority class using ADASYN may lead to overfitting, as it often involves the creation of exact copies or near-replicas of minority class instances. This practice can also risk introducing noise, especially if synthetic samples are generated without consideration of the underlying data distribution.

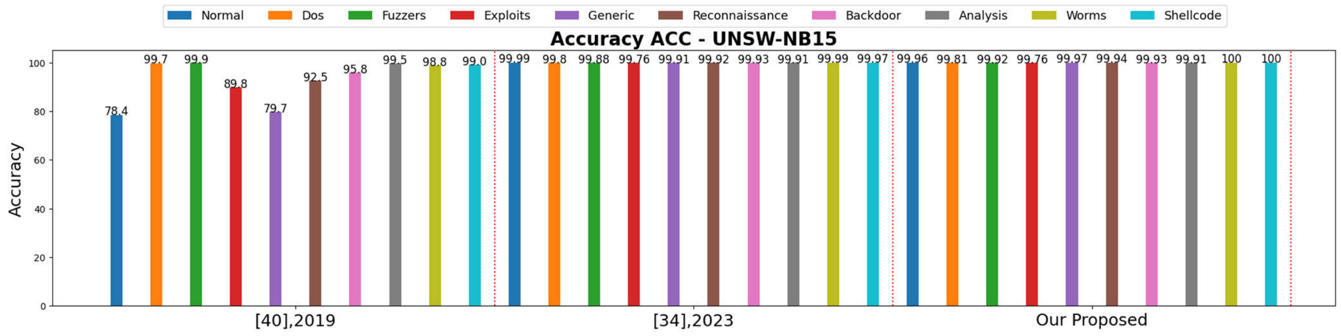


FIGURE 11. Comparison of the performance between our approach and previous research in terms of ACC for each category in the UNSW-NB15.

TABLE 14. Comparison of the performance between our approach and previous research in terms of recall for each category in the UNSW-NB15.

Class	[66] R, 2017	[68] R, 2017	[76] R, 2018	[78] R, 2019	[40] R, 2019	[86] R, 2020	[34] R, 2023	Our Proposal
								R
Normal	97.38	96.30	70.30	82.00	94.70	100.00	100.00	99.95
DoS	84.81	35.70	69.30	0.44	97.80	12.63	81.90	85.80
Fuzzers	87.50	28.90	60.70	40.30	00.00	83.86	92.74	94.91
Exploits	95.61	72.80	60.70	57.14	00.17	89.44	93.03	94.25
Generic	97.81	97.60	96.50	61.21	57.70	97.33	99.37	99.67
Reconnaissance	83.80	80.60	83.70	24.89	04.50	66.61	91.90	91.35
Backdoor	70.44	02.10	16.00	00.00	33.55	63.94	73.91	76.99
Analysis	69.83	00.90	17.40	1.34	00.00	12.13	78.57	78.51
Worms	38.24	75.00	90.90	00.00	00.00	24.64	85.71	100
Shellcode	58.20	29.10	69.30	00.85	00.00	36.51	89.19	100

TABLE 15. Comparison of the performance between our approach and previous research in terms of FAR for each category in the UNSW-NB15.

Class	[68] FAR, 2017	[78] FAR, 2019	[40] FAR, 2019	[34] FAR, 2023	Our Proposal
					FAR
Normal	0.106	0.001	0.299	0.000	0.000
DoS	0.023	5.478	0.000	0.082	0.098
Fuzzers	0.017	1.321	0.000	0.050	0.028
Exploits	0.073	1.391	0.000	0.119	0.145
Generic	0.005	0.519	0.155	0.034	0.001
Reconnaissance	0.004	1.438	0.000	0.037	0.013
Backdoor	0.000	0.822	0.000	0.048	0.052
Analysis	0.000	0.789	0.000	0.065	0.067
Worms	0.000	0.062	0.000	0.003	0.000
Shellcode	0.002	0.435	0.000	0.028	0.000

Conversely, under-sampling the majority class through RUS might result in a loss of information by removing instances from the dataset that could have been crucial for the model's learning. However, according to our findings, particularly the curves depicting significant convergence between training and validation sets, these potential issues appear to have a very slight, almost negligible, effect.

4. Random Forest models can be relatively slow in generating predictions compared to other algorithms, particularly when dealing with datasets of high dimensionality or when the number of trees within the model is large. Despite being recognized as a robust model, it may still succumb to overfitting if not properly tuned. Additionally, understanding the decision-making process in RF model can be challenging, as they often function as a "black box," rendering the logic behind their decisions somewhat opaque and difficult to interpret. However, our performance evaluations have shown superiority compared to other classifiers, particularly after determining the optimal number of trees based on the empirical results.
5. Evaluation Datasets: Creating and maintaining an up-to-date dataset for IDS is both challenging and costly. IDS must grapple with the constant evolution of novel attacks, network anomalies, and previously unseen outliers. Consequently, stability becomes a concern, requiring the model to adapt continually to changes in networks and threats. Specific evaluation datasets may not generalize to other contexts and might contain biases or a lack of

TABLE 16. Evaluation metrics across all classes in the CIC-DDoS 2019 dataset.

Category	ACC	R	FAR	P	F	MAR
Syn	1.00	99.99	0.0000	1.00	1.00	0.0080
TFTP	1.00	1.00	0.0016	99.98	99.99	0.0000
LDAP	1.00	99.96	0.0000	1.00	99.98	0.0399
UDP	1.00	99.96	0.0008	99.99	99.98	0.0398
SNMP	1.00	1.00	0.0000	1.00	1.00	0.0000
SSDP	1.00	99.96	0.0079	99.92	99.94	0.0401
NetBIOS	1.00	99.98	0.0000	1.00	99.99	0.0238
MSSQL	99.99	99.92	0.0007	99.99	99.95	0.0825
DNS	99.99	99.97	0.0112	99.88	99.93	0.0332
UDP-lag	99.99	99.97	0.0024	99.98	99.97	0.0329
NTP	1.00	1.00	0.0039	99.95	99.97	0.0000
BENIGN	1.00	1.00	0.0007	99.98	99.99	0.0000
WebDDoS	1.00	1.00	0.0000	1.00	1.00	0.0000

TABLE 17. Performance comparison of our proposal with other studies, based on the recall metric for each class in the CIC-DDoS 2019 dataset.

Category	[31] R, 2021	[88] R, 2021	[95] R, 2022	Our Study (R)
Syn	92.43	1.00	99.88	99.99
TFTP	86.16	-	99.25	1.00
LDAP	99.88	72.7	-	99.96
UDP	98.21	71.9	99.22	99.96
SNMP	99.55	-	99.84	1.00
SSDP	99.79	92.3	98.54	99.96
NetBIOS	99.90	94.1	99.63	99.98
MSSQL	99.10	38.6	98.61	99.92
DNS	97.21	74.4	99.72	99.97
UDP-lag	98.67	87.2	97.83	99.97
NTP	97.12	95.4	99.80	1.00
BENIGN	-	99.5	99.99	1.00
WebDDoS	99.32	-	98.86	1.00

TABLE 18. Performance comparison of our proposal with other studies, based on the FPR and ACC metrics for each class in the CIC-DDoS 2019 dataset.

Category	[95] FPR, 2022	Our Study (FPR)	[31] ACC, 2021	Our Study (ACC)
Syn	0.0000	0.0000	99.69	1.00
TFTP	0.0009	0.0016	99.83	1.00
LDAP	-	0.0000	99.95	1.00
UDP	0.0010	0.0008	99.69	1.00
SNMP	0.0012	0.0000	99.97	1.00
SSDP	0.0015	0.0079	99.91	1.00
NetBIOS	0.0002	0.0000	99.94	1.00
MSSQL	0.0012	0.0007	99.82	99.99
DNS	0.0002	0.0112	99.51	99.99
UDP-lag	0.0009	0.0024	99.87	99.99
NTP	0.0000	0.0039	99.52	1.00
BENIGN	0.0000	0.0007	-	1.00
WebDDoS	0.0005	0.0000	95.11	1.00

TABLE 19. Evaluation metrics across all classes in the CIC Bell DNS EXF 2021 dataset.

Category	ACC	R	FAR	P	F	MAR
Heavy-Benign	94.78	91.95	3.685	93.14	92.54	8.048
Light-Benign	96.77	72.33	0.000	1.00	83.94	27.67
Heavy Attack	94.60	99.78	9.712	89.56	94.39	0.224
Light Attack	98.42	79.68	0.033	99.50	88.49	20.32

diversity that could skew results. However, in the current work, we utilized the most recent datasets to test and validate our proposal. These datasets are diverse and

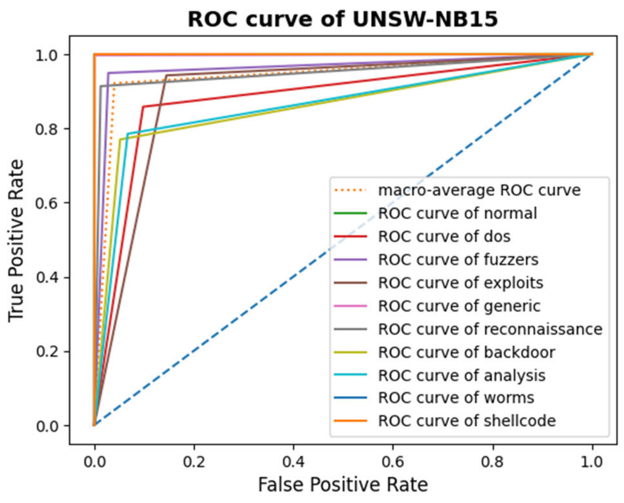


FIGURE 12. ROC curve of the UNSW-NB15 dataset.

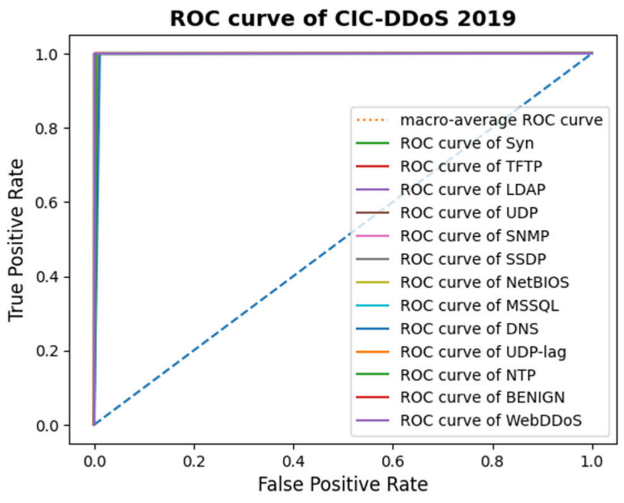


FIGURE 13. ROC curve of the CIC-DDoS 2019 dataset.

encompass the important instances of new attacks on real-world network environments that exist today. Moreover, by using a variety of performance measures on these datasets and achieving satisfactory results compared to other studies, we demonstrated the robustness and generalizability of our model.

6. Real-world Implementation Challenges: Practical implementation can reveal unexpected challenges when transitioning from theoretical or lab-tested proposals to real-world environments. These challenges may include unanticipated problems such as integration issues with existing systems, performance deficiencies under real-world loads, unforeseen security vulnerabilities, and inefficiencies outside of a controlled lab setting. To mitigate these uncertainties, it is advisable to test the proposal in a real-time setting following lab evaluation. Doing so ensures that the strategy is tailored for real-world application, a process we plan to pursue in our future work.

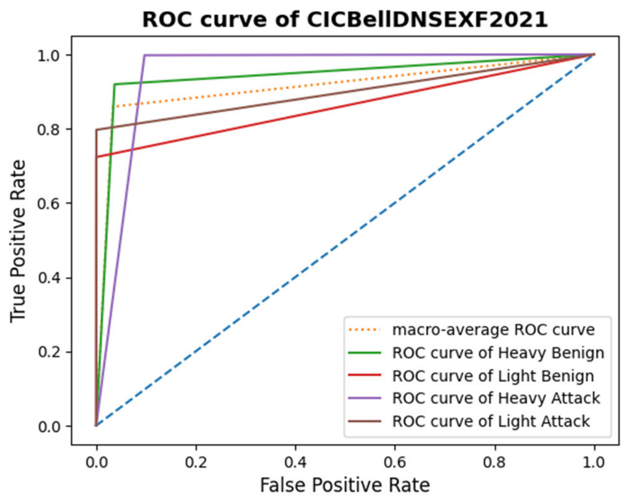


FIGURE 14. ROC curve of the CIC Bell DNS EXF 2021 dataset.

VI. CONSIDERATIONS FOR PRACTICAL IMPLEMENTATION

In the context of applying our proposal in a real-world environment (live testing), we recognize, as previously noted, that this remains a current limitation of our study. Nevertheless, this section outlines key considerations that will facilitate the practical implementation of our approach in future endeavors:

- Parallelizability: The architecture of our approach, specifically the components that necessitate iterating over each grasshopper or feature, is aptly designed for parallel computation. This design can substantially enhance runtime efficiency, a crucial factor in real-world cloud computing environments. These environments often feature modern multi-core processors and leverage distributed computing techniques, making our approach particularly well-suited for such settings
- Batch Processing or Sampling: Considering the potentially substantial number of instances in our datasets, we may suggest employing batch processing or sampling techniques, particularly within cloud settings. These methods are highly effective at significantly reducing the computational burden during fitness evaluations. The availability and scalability of cloud infrastructures provide an efficient platform for the implementation of these techniques, thereby enhancing the feasibility of our approach when processing large datasets.
- Hardware Resource Considerations: We emphasize the importance of securing adequate computational resources, particularly for the effective processing of large-scale datasets. This challenge is more manageable in a cloud environment, where the 'pay-as-you-use' model provides scalable resources tailored to specific computational needs. Such an environment facilitates flexible and cost-effective access to the necessary hard-

TABLE 20. The abbreviation table.

Nomenclature	Abbreviation
IDS	Intrusion Detection System
CC	Cloud Computing
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
EL	Ensemble Learning
DT	Decision Tree
RF	Random Forest
XGB/ XGBoost	eXtreme Gradient Boosting
CART	Classification and Regression Trees
ID3	Iterative Dichotomiser 3
NB-Tree	Naive Bayes Tree
GOA	Grasshopper Optimization Algorithm
GA	Genetic Algorithm
Ramp-KSVCR	Ramp Loss K-Support Vector Classification-Regression
A-PCA	Adaptive - Principal Component Analysis
GMM	Gaussian Mixture Models
Weka-ML	It's open-source ML software used for feature selection
CS-RFE	Chi Square - Recursive Feature Elimination
GAA-ADS	Geometric Area Analysis-Anomaly-based Detection
PSO	Particle Swarm Optimization
DBN	Deep Belief Network
CFS	Correlation-based Feature Selection
ABC-AFS	Artificial Bee Colony-Artificial Fish Swarm
FI	Feature Importance model
FSFF	Feature Similarity-based Fitness Function
CAFF	Classifier Accuracy based Fitness Function
TSDL (D-SAE)	Two-Stage Deep Learning (Deep Stacked Auto-Encoder)
IC-VAE	Improved Conditional Variational Autoencoder
ACO	Ant Colony Optimization
GWO	Grey Wolf Optimizer
FFA	Firefly Optimization
IG	Information Gain
NSGA2	Non-Dominated Sorting Genetic Algorithm 2
MLR	Multinomial LR
LR	Logistic Regression
MSCNN	Multi-Scale Convolutional Neural Networks
WFEU	Wrapper-based Feature Extraction Unit
FE	Feature Extraction
FS	Features Selection
SVM	Support Vector Machine
CAE+OCSVM	Convolutional Autoencoder + One-Class SVM
LSO	Lion Swarm Optimization
TS	Tabu Search
MFFSEM	Multi-Dimensional Feature Fusion And Stacking Ensemble Mechanism
BWO	Black Widow Optimizer
SOA	Sandpiper Optimization Algorithm
EEDTL	Extended Equilibrium Deep Transfer Learning
BBA	Binary Bat Algorithm
DDoS	Distributed Denial of Service
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
MSSQL	Microsoft SQL Server
SSDP	Simple Service Discovery Protocol
NTP	Network Time Protocol
TFTP	Trivial FileTransfer Protocol
DNS	Domain Name Server
LDAP	Lightweight Directory Access Protocol
NETBIOS	Network Basic Input/Output System
SNMP	Simple Network Management Protocol

TABLE 20. ***(Continued.)*** **The abbreviation table.**

SYN	Synchronize
EXF	Exfiltration
SMOTE	Synthetic Minority Oversampling Technique
ADASYN	Adaptive Synthetic
RUS	Random Under-Sampling
HL	Hidden Layer
ANN	Artificial Neural Network
ELM	Extreme Learning Machine
I-ELM	Incremental-Extreme Learning Machine
FLN	Fast Learning Network. This model is an updated version of ANN and ELM, which overcomes their weaknesses.
LSTM	Long Short-Term Memory is one of the common types of Recurrent Neural Networks (RNNs)
Bi-LSTM	Bidirectional Long Short-Term Memory
HMLSTM	Hierarchical Multi-Scale LSTM
GRU	Gated Recurrent Unit
AlexNet	Is a common type of Convolutional Neural Network
KNN	K-Nearest Neighbors
DNN	Deep Neural Network
FFDNN	Feed-Forward Deep Neural Network
ISSA	Improved Salp Swarm Algorithm
LASSO	Least Absolute Shrinkage and Selection Operator
MI	Mutual Information
LGB	Light Gradient Boost
AB	AdaBoost
MLP	Multilayer Perceptron
SNN	Simple Neural Network
WMV	Weighted Majority Voting
BSO	Brain Storm Optimization
HGSO	Henry's Gas Solubility Optimization
RFECV	Recursive Feature Elimination
SHAP	Shapley Additive Explanations
KNORA-U	K-Nearest Oracles Union
HOSVD	Higher Order Singular Value Decomposition

TABLE 21. **Corresponding data types for each feature in the UNSW-NB15 dataset.**

No	Feature Name	Type	No	Feature Name	Type
0	srcip	object	**25**	res_bdy_len	int64
1	sport	int64	**26**	sjit	float64
2	dstip	object	**27**	djit	float64
3	dsport	int64	**28**	stime	int64
4	proto	object	**29**	ltime	int64
5	state	object	**30**	sintpkt	float64
6	dur	float64	**31**	dintpkt	float64
7	sbytes	int64	**32**	tcprtt	float64
8	dbytes	int64	**33**	synack	float64
9	sttl	int64	**34**	ackdat	float64
10	dttl	int64	**35**	is_sm_ips_ports	int64
11	sloss	int64	**36**	ct_state_ttl	int64
12	dloss	int64	**37**	ct_flw_http_mthd	float64
13	service	object	**38**	is_ftp_login	int64
14	sload	float64	**39**	ct_ftp_cmd	int64
15	dload	float64	**40**	ct_srv_src	int64
16	spkts	int64	**41**	ct_srv_dst	int64
17	dpkts	int64	**42**	ct_dst_ltm	int64
18	swin	int64	**43**	ct_src_ltm	int64
19	dwin	int64	**44**	ct_src_dport_ltm	int64
20	stcpb	int64	**45**	ct_dst_sport_ltm	int64
21	dtcpb	int64	**46**	ct_dst_src_ltm	int64
22	smeansz	int64	**47**	attack_cat	object
23	dmeansz	int64	**48**	label	int64
24	trans_depth	int64			

ware resources, thereby enhancing the feasibility of processing extensive data sets efficiently.

The aforementioned considerations are intended to establish a preliminary foundation for future live testing and the practical deployment of our approach.

VII. CONCLUSION

Security in the cloud is of paramount importance due to the increasing use of cloud computing by both individuals and organizations. To detect threats and protect user data, machine learning classifiers were employed to categorize network packets as either benign or malicious. The proposed intrusion detection system aims to harness enhanced intrusion detection accuracy by merging the advantages of two feature selection algorithms, GA and GOA. This combination identifies an optimal feature subset and leverages the exploitation capacity of GA and the exploration capabilities of GOA. The operation of GA prevents GOA from getting stuck in local solutions. This not only augments the approach's performance but also conserves computational resources. The hybrid feature selection method selects the best features, while the classifiers categorize the network data. Demonstrating a high recall and a lower FAR, the proposal proved proficient in identifying various types of attacks. The evaluation was conducted based on accuracy, recall, FAR, precision, f-measure, MAR, and ROC curves. Different classification algorithms were tested on the proposed model, and the evaluation results confirmed that the random forest classifier outperformed the others across all three datasets. While there are numerous studies in the literature utilizing meta-heuristic algorithms for selecting optimal features or for optimizing the weights of neural network classifiers, our proposal's findings surpassed the performance of the majority of these methodologies.

In future work, we intend to leverage optimal features derived from hybrid metaheuristic algorithms, such as GOA-GA, to fine-tune and optimize the weights of neural network models. Our objective is to enhance the efficacy of neural network classifiers by adjusting parameters, including the number of hidden layers, connection density, choice of activation functions, and the number of epochs. Furthermore, we aim to propose a systematic hyperparameter tuning of the classifiers, employing techniques such as grid search, random search, and metaheuristic algorithms. This future endeavor will include a comprehensive comparative analysis of the results to identify the most effective tuning strategy. An additional focus will be to devise a robust methodology to address issues related to data imbalance.

APPENDIX A

Table 20 showcases the abbreviations utilized within the manuscript, which are not detailed in their full forms. They are arranged in the sequence in which they appear.

TABLE 22. Specification of data types for all features in the CIC-DDoS2019 dataset.

No	Feature Name	Type	No	Feature Name	Type
0	Unnamed: 0	int64	**44**	Bwd Packets/s	float64
1	Flow ID	object	**45**	Min Packet Length	float64
2	Source IP	object	**46**	Max Packet Length	float64
3	Source Port	int64	**47**	Packet Length Mean	float64
4	Destination IP	object	**48**	Packet Length Std	float64
5	Destination Port	int64	**49**	Packet Length Variance	float64
6	Protocol	int64	**50**	FIN Flag Count	int64
7	Timestamp	object	**51**	SYN Flag Count	int64
8	Flow Duration	int64	**52**	RST Flag Count	int64
9	Total Fwd Packets	int64	**53**	PSH Flag Count	int64
10	Total Backward Packets	int64	**54**	ACK Flag Count	int64
11	Total Length of Fwd Packets	float64	**55**	URG Flag Count	int64
12	Total Length of Bwd Packets	float64	**56**	CWE Flag Count	int64
13	Fwd Packet Length Max	float64	**57**	ECE Flag Count	int64
14	Fwd Packet Length Min	float64	**58**	Down/Up Ratio	float64
15	Fwd Packet Length Mean	float64	**59**	Average Packet Size	float64
16	Fwd Packet Length Std	float64	**60**	Avg Fwd Segment Size	float64
17	Bwd Packet Length Max	float64	**61**	Avg Bwd Segment Size	float64
18	Bwd Packet Length Min	float64	**62**	Fwd Header Length.1	int64
19	Bwd Packet Length Mean	float64	**63**	Fwd Avg Bytes/Bulk	int64
20	Bwd Packet Length Std	float64	**64**	Fwd Avg Packets/Bulk	int64
21	Flow Bytes/s	float64	**65**	Fwd Avg Bulk Rate	int64
22	Flow Packets/s	float64	**66**	Bwd Avg Bytes/Bulk	int64
23	Flow IAT Mean	float64	**67**	Bwd Avg Packets/Bulk	int64
24	Flow IAT Std	float64	**68**	Bwd Avg Bulk Rate	int64
25	Flow IAT Max	float64	**69**	Subflow Fwd Packets	int64
26	Flow IAT Min	float64	**70**	Subflow Fwd Bytes	int64
27	Fwd IAT Total	float64	**71**	Subflow Bwd Packets	int64
28	Fwd IAT Mean	float64	**72**	Subflow Bwd Bytes	int64
29	Fwd IAT Std	float64	**73**	Init_Win_bytes_forward	int64
30	Fwd IAT Max	float64	**74**	Init_Win_bytes_backward	int64
31	Fwd IAT Min	float64	**75**	act_data_pkt_fwd	int64
32	Bwd IAT Total	float64	**76**	min_seg_size_forward	int64
33	Bwd IAT Mean	float64	**77**	Active Mean	float64
34	Bwd IAT Std	float64	**78**	Active Std	float64
35	Bwd IAT Max	float64	**79**	Active Max	float64
36	Bwd IAT Min	float64	**80**	Active Min	float64
37	Fwd PSH Flags	int64	**81**	Idle Mean	float64
38	Bwd PSH Flags	int64	**82**	Idle Std	float64
39	Fwd URG Flags	int64	**83**	Idle Max	float64
40	Bwd URG Flags	int64	**84**	Idle Min	float64
41	Fwd Header Length	int64	**85**	SimillarHTTP	object
42	Bwd Header Length	int64	**86**	Inbound	int64
43	Fwd Packets/s	float64	**87**	Label	object

TABLE 22. *(Continued.)* Specification of data types for all features in the CIC-DDoS2019 dataset.

TABLE 23. Data type allocation for each feature in the CIC Bell DNS EXF 2021 dataset.

No		Feature Name	Type	No		Feature Name	Type
0	Stateful Features	rr	float64	**22**	Stateful Features	reverse_dns	object
1		A_frequency	float64	**23**		a_records	float64
2		NS_frequency	float64	**24**		unique_ttl	object
3		CNAME_frequency	float64	**25**		ttl_mean	float64
4		SOA_frequency	float64	**26**		ttl_variance	float64
5		NULL_frequency	float64	**27**	label		object
6		PTR_frequency	float64	**28**	timestamp		object
7		HINFO_frequency	float64	**29**	Stateless Features	FQDN_count	float64
8		MX_frequency	float64	**30**		subdomain_length	float64
9		TXT_frequency	float64	**31**		upper	float64
10		AAAA_frequency	float64	**32**		lower	float64
11		SRV_frequency	float64	**33**		numeric	float64
12		OPT_frequency	float64	**34**		entropy	float64
13		rr_type	object	**35**		special	float64
14		rr_count	float64	**36**		labels	float64
15		rr_name_entropy	float64	**37**		labels_max	float64
16		rr_name_length	float64	**38**		labels_average	float64
17		distinct_ns	float64	**39**		longest_word	object
18		distinct_ip	object	**40**		sld	object
19		unique_country	object	**41**		len	float64
20		unique_asn	object	**42**		subdomain	float64
21		distinct_domains	object				

APPENDIX B

Tables 21, 22, and 23 detail every attribute, along with its corresponding data type, in the public datasets UNSW-NB15, CIC-DDoS2019, and CIC Bell DNS EXF 2021, respectively. These data types were identified by consulting the descriptions of each dataset from their official sources, as cited in Table 1, and also by employing the info() method offered by the Pandas library.

ACKNOWLEDGMENT

The authors wish to extend their sincere gratitude to the Deanship of Scientific Research, Islamic University of Madinah, for the support provided to the Post-Publishing Program 3.

REFERENCES

[1] R. R. Kumar, A. Tomar, M. Shameem, and M. N. Alam, ''OPTCLOUD: An optimal cloud service selection framework using QoS correlation lens,'' *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–16, May 2022, doi: 10.1155/2022/2019485.

[2] R. R. Kumar, M. Shameem, and C. Kumar, ''A computational framework for ranking prediction of cloud services under fuzzy environment,'' *Enterprise Inf. Syst.*, vol. 16, no. 1, pp. 167–187, Jan. 2022, doi: 10.1080/17517575.2021.1889037.

[3] M. A. Akbar, M. Shameem, S. Mahmood, A. Alsanad, and A. Gumaei, ''Prioritization based taxonomy of cloud-based outsource software development challenges: Fuzzy AHP analysis,'' *Appl. Soft Comput.*, vol. 95, Oct. 2020, Art. no. 106557, doi: 10.1016/j.asoc.2020.106557.

[4] M. Bakro, S. K. Bisoy, A. K. Patel, and M. A. Naal, ''Performance analysis of cloud computing encryption algorithms,'' in *Advances in Intelligent Computing and Communication* (Lecture Notes in Networks and Systems), vol. 202, 2021, pp. 357–367, doi: 10.1007/978-981-16-0695-3_35.

[5] *Cyber Security Market Share, Forecast | Growth Analysis [2030]*. Accessed: Apr. 23, 2023. [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/cyber-security-market-101165

[6] M. Rashid, J. Kamruzzaman, T. Imam, S. Wibowo, and S. Gordon, ''A tree-based stacking ensemble technique with feature selection for network intrusion detection,'' *Int. J. Speech Technol.*, vol. 52, no. 9, pp. 9768–9781, Jul. 2022, doi: 10.1007/s10489-021-02968-1.

[7] M. Bakro, S. K. Bisoy, A. K. Patel, and M. A. Naal, ''Hybrid blockchain-enabled security in cloud storage infrastructure using ECC and AES algorithms,'' in *Blockchain Based Internet of Things* (Lecture Notes on Data Engineering and Communications Technologies), vol. 112. Singapore: Springer, 2022, pp. 139–170.

[8] P. Ghosh, S. Sinha, R. R. Sharma, and S. Phadikar, ''An efficient IDS in cloud environment using feature selection based on DM algorithm,'' *J. Comput. Virol. Hacking Techn.*, vol. 18, no. 3, pp. 243–258, Sep. 2022, doi: 10.1007/s11416-021-00410-1.

[9] P. R. Kanna and P. Santhi, ''Hybrid intrusion detection using MapReduce based black widow optimized convolutional long short-term memory neural networks,'' *Expert Syst. Appl.*, vol. 194, May 2022, Art. no. 116545, doi: 10.1016/j.eswa.2022.116545.

[10] S. Cai, D. Han, X. Yin, D. Li, and C.-C. Chang, ''A hybrid parallel deep learning model for efficient intrusion detection based on metric learning,'' *Connection Sci.*, vol. 34, no. 1, pp. 551–577, Dec. 2022, doi: 10.1080/09540091.2021.2024509.

[11] A. Binbusayyis and T. Vaiyapuri, ''Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class SVM,'' *Int. J. Speech Technol.*, vol. 51, no. 10, pp. 7094–7108, Oct. 2021, doi: 10.1007/s10489-021-02205-9.

[12] P. R. Kannari, N. C. Shariff, and R. L. Biradar, ''Network intrusion detection using sparse autoencoder with swish-PReLU activation model,'' *J. Ambient Intell. Humanized Comput.*, Mar. 2021, doi: 10.1007/s12652-021-03077-0.

[13] A. K. Naik, V. Kuppili, and D. R. Edla, ''Efficient feature selection using one-pass generalized classifier neural network and binary bat algorithm with a novel fitness function,'' *Soft Comput.*, vol. 24, no. 6, pp. 4575–4587, Mar. 2020, doi: 10.1007/s00500-019-04218-6.

[14] V. R. Balasaraswathi, M. Sugumaran, and Y. Hamid, ''Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms,'' *J. Commun. Inf. Netw.*, vol. 2, no. 4, pp. 107–119, Dec. 2017, doi: 10.1007/s41650-017-0033-7.

[15] M. H. Nasir, S. A. Khan, M. M. Khan, and M. Fatima, ''Swarm intelligence inspired intrusion detection systems—A systematic literature review,'' *Comput. Netw.*, vol. 205, Mar. 2022, Art. no. 108708, doi: 10.1016/j.comnet.2021.108708.

[16] V. R. S. Dora and V. N. Lakshmi, ''Optimal feature selection with CNN-feature learning for DDoS attack detection using meta-heuristic-based LSTM,'' *Int. J. Intell. Robot. Appl.*, vol. 6, no. 2, pp. 323–349, Jun. 2022, doi: 10.1007/s41315-022-00224-4.

[17] P. R. Kanna and P. Santhi, ''Unified deep learning approach for efficient intrusion detection system using integrated spatial–temporal features,'' *Knowl.-Based Syst.*, vol. 226, Aug. 2021, Art. no. 107132, doi: 10.1016/j.knosys.2021.107132.

[18] T. Kim and W. Pak, ''Real-time network intrusion detection using deferred decision and hybrid classifier,'' *Future Gener. Comput. Syst.*, vol. 132, pp. 51–66, Jul. 2022, doi: 10.1016/j.future.2022.02.011.

[19] S. Dwivedi, M. Vardhan, and S. Tripathi, ''Building an efficient intrusion detection system using grasshopper optimization algorithm for anomaly detection,'' *Cluster Comput.*, vol. 24, no. 3, pp. 1881–1900, Sep. 2021, doi: 10.1007/s10586-020-03229-5.

[20] A. A. Süzen, ''Developing a multi-level intrusion detection system using hybrid-DBN,'' *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 2, pp. 1913–1923, Feb. 2021, doi: 10.1007/s12652-020-02271-w.

[21] P. J. Sajith and G. Nagarajan, ''Intrusion detection system using deep belief network & particle swarm optimization,'' *Wireless Pers. Commun.*, vol. 125, no. 2, pp. 1385–1403, Jul. 2022, doi: 10.1007/s11277-022-09609-x.

[22] G. Sreelatha, A. V. Babu, and D. Midhunchakkaravarthy, ''Improved security in cloud using sandpiper and extended equilibrium deep transfer learning based intrusion detection,'' *Cluster Comput.*, vol. 25, no. 5, pp. 3129–3144, Oct. 2022, doi: 10.1007/s10586-021-03516-9.

[23] Q. Liu, D. Wang, Y. Jia, S. Luo, and C. Wang, ''A multi-task based deep learning approach for intrusion detection,'' *Knowl.-Based Syst.*, vol. 238, Feb. 2022, Art. no. 107852, doi: 10.1016/j.knosys.2021.107852.

[24] M. Moizuddin and M. V. Jose, ''A bio-inspired hybrid deep learning model for network intrusion detection,'' *Knowl.-Based Syst.*, vol. 238, Feb. 2022, Art. no. 107894, doi: 10.1016/j.knosys.2021.107894.

[25] C.-M. Hsu, M. Z. Azhari, H.-Y. Hsieh, S. W. Prakosa, and J.-S. Leu, ''Robust network intrusion detection scheme using long-short term memory based convolutional neural networks,'' *Mobile Netw. Appl.*, vol. 26, no. 3, pp. 1137–1144, Jun. 2021, doi: 10.1007/s11036-020-01623-2.

[26] P. Ghosh, Z. Alam, R. R. Sharma, and S. Phadikar, ''An efficient SGM based IDS in cloud environment,'' *Computing*, vol. 104, no. 3, pp. 553–576, Mar. 2022, doi: 10.1007/s00607-022-01059-4.

[27] E.-U.-H. Qazi, M. Imran, N. Haider, M. Shoaib, and I. Razzak, ''An intelligent and efficient network intrusion detection system using deep learning,'' *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107764, doi: 10.1016/j.compeleceng.2022.107764.

[28] S. K. Gupta, M. Tripathi, and J. Grover, ''Hybrid optimization and deep learning based intrusion detection system,'' *Comput. Electr. Eng.*, vol. 100, May 2022, Art. no. 107876, doi: 10.1016/j.compeleceng.2022.107876.

[29] E. Balamurugan, A. Mehbodniya, E. Kariri, K. Yadav, A. Kumar, and M. Anul Haq, ''Network optimization using defender system in cloud computing security based intrusion detection system withgame theory deep neural network (IDSGT-DNN),'' *Pattern Recognit. Lett.*, vol. 156, pp. 142–151, Apr. 2022, doi: 10.1016/j.patrec.2022.02.013.

[30] R. Du, Y. Li, X. Liang, and J. Tian, ''Support vector machine intrusion detection scheme based on cloud-fog collaboration,'' *Mobile Netw. Appl.*, vol. 27, no. 1, pp. 431–440, Feb. 2022, doi: 10.1007/s11036-021-01838-x.

[31] S. U. Rehman, M. Khaliq, S. I. Imtiaz, A. Rasool, M. Shafiq, A. R. Javed, Z. Jalil, and A. K. Bashir, ''DIDDOS: An approach for detection and identification of distributed denial of service (DDoS) cyberattacks using gated recurrent units (GRU),'' *Future Gener. Comput. Syst.*, vol. 118, pp. 453–466, May 2021, doi: 10.1016/j.future.2021.01.022.

[32] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, ''AE-MLP: A hybrid deep learning approach for DDoS detection and classification,'' *IEEE Access*, vol. 9, pp. 146810–146821, 2021, doi: 10.1109/ACCESS.2021.3123791.

[33] R. Patil, H. Dudeja, and C. Modi, ''Designing an efficient security framework for detecting intrusions in virtual network of cloud computing,'' *Comput. Secur.*, vol. 85, pp. 402–422, Aug. 2019, doi: 10.1016/j.cose.2019.05.016.

[34] M. Bakro, R. R. Kumar, A. Alabrah, Z. Ashraf, M. N. Ahmed, M. Shameem, and A. Abdelsalam, "An improved design for a cloud intrusion detection system using hybrid features selection approach with ML classifier," *IEEE Access*, vol. 11, pp. 64228–64247, 2023, doi: 10.1109/ACCESS.2023.3289405.

[35] A. Alabrah, "An efficient NIDPS with improved salp swarm feature optimization method," *Appl. Sci.*, vol. 13, no. 12, p. 7002, Jun. 2023, doi: 10.3390/app13127002.

[36] *The UNSW-NB15 Dataset | UNSW Research*. Accessed: Jul. 18, 2023. [Online]. Available: https://research.unsw.edu.au/projects/unsw-nb15-dataset

[37] (2019). *DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. Accessed: Jul. 18, 2023. [Online]. Available: https://www.unb.ca/cic/datasets/ddos-2019.html

[38] *CIC-Bell-DNS-EXF 2021 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. Accessed: Jul. 18, 2023. [Online]. Available: https://www.unb.ca/cic/datasets/dns-exf-2021.html

[39] 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)*, 2015, doi: 10.1109/MilCIS.2015.7348942.

[40] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[41] O. Almomani, "A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms," *Symmetry*, vol. 12, no. 6, pp. 1–20, Jun. 2020, doi: 10.3390/sym12061046.

[42] C. Khammassi and S. Krichen, "A NSGA2-LR wrapper approach for feature selection in network intrusion detection," *Comput. Netw.*, vol. 172, May 2020, Art. no. 107183, doi: 10.1016/j.comnet.2020.107183.

[43] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8, doi: 10.1109/CCST.2019.8888419.

[44] A. E. Cil, K. Yildiz, and A. Buldu, "Detection of DDoS attacks with feed forward based deep neural network model," *Expert Syst. Appl.*, vol. 169, May 2021, Art. no. 114520, doi: 10.1016/j.eswa.2020.114520.

[45] R. K. Batchu and H. Seetha, "On improving the performance of DDoS attack detection system," *Microprocessors Microsystems*, vol. 93, Sep. 2022, Art. no. 104571, doi: 10.1016/j.micpro.2022.104571.

[46] R. K. Batchu and H. Seetha, "A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning," *Comput. Netw.*, vol. 200, Dec. 2021, Art. no. 108498, doi: 10.1016/j.comnet.2021.108498.

[47] K. Žiža, P. Tadić, and P. Vuletić, "DNS exfiltration detection in the presence of adversarial attacks and modified exfiltrator behaviour," *Int. J. Inf. Secur.*, vol. 22, no. 6, pp. 1865–1880, Dec. 2023, doi: 10.1007/s10207-023-00723-w.

[48] S. Wang, L. Sun, S. Qin, W. Li, and W. Liu, "KRTunnel: DNS channel detector for mobile devices," *Comput. Secur.*, vol. 120, Sep. 2022, Art. no. 102818, doi: 10.1016/j.cose.2022.102818.

[49] S. Garcia and V. Valeros, "Towards a better labeling process for network security datasets," May 2023, *arXiv:2305.01337*. Accessed: Jul. 26, 2023.

[50] S. Mahdavifar, A. H. Salem, P. Victor, A. H. Razavi, M. Garzon, N. Hellberg, and A. H. Lashkari, "Lightweight hybrid detection of data exfiltration using DNS based on machine learning," in *Proc. 11th Int. Conf. Commun. Netw. Secur.*, Dec. 2021, pp. 80–86, doi: 10.1145/3507509.3507520.

[51] K. Potdar, T. S. Pardawala, and C. D. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," *Int. J. Comput. Appl.*, vol. 175, no. 4, pp. 7–9, Oct. 2017, doi: 10.5120/ijca2017915495.

[52] S. Afreen, A. K. Bhurjee, and R. M. Aziz, "Gene selection with game Shapley Harris hawks optimizer for cancer classification," *Chemometric Intell. Lab. Syst.*, vol. 242, Nov. 2023, Art. no. 104989, doi: 10.1016/j.chemolab.2023.104989.

[53] M. C. V. Suresh and J. B. Edward, "A hybrid algorithm based optimal placement of DG units for loss reduction in the distribution system," *Appl. Soft Comput.*, vol. 91, Jun. 2020, Art. no. 106191, doi: 10.1016/j.asoc.2020.106191.

[54] Y. Meraihi, A. B. Gabis, S. Mirjalili, and A. Ramdane-Cherif, "Grasshopper optimization algorithm: Theory, variants, and applications," *IEEE Access*, vol. 9, pp. 50001–50024, 2021, doi: 10.1109/ACCESS.2021.3067597.

[55] R. M. Aziz, "Nature-inspired metaheuristics model for gene selection and classification of biomedical microarray data," *Med. Biol. Eng. Comput.*, vol. 60, no. 6, pp. 1627–1646, Jun. 2022, doi: 10.1007/s11517-022-02555-7.

[56] L. Abualigah and A. J. Dulaimi, "A novel feature selection method for data mining tasks using hybrid sine cosine algorithm and genetic algorithm," *Cluster Comput.*, vol. 24, no. 3, pp. 2161–2176, Sep. 2021, doi: 10.1007/s10586-021-03254-y.

[57] S. Harifi, J. Mohammadzadeh, M. Khalilian, and S. Ebrahimnejad, "Hybrid-EPC: An emperor penguins colony algorithm with crossover and mutation operators and its application in community detection," *Prog. Artif. Intell.*, vol. 10, no. 2, pp. 181–193, Jun. 2021, doi: 10.1007/s13748-021-00231-9.

[58] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," *IEEE Access*, vol. 9, pp. 7550–7563, 2021, doi: 10.1109/ACCESS.2020.3048198.

[59] X. Ma and W. Shi, "AESMOTE: Adversarial reinforcement learning with SMOTE for anomaly detection," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 943–956, Apr. 2021, doi: 10.1109/TNSE.2020.3004312.

[60] 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, doi: 10.1109/ACCESS.2020.2973730.

[61] A. K. Verma, P. Kaushik, and G. Shrivastava, "A network intrusion detection approach using variant of convolution neural network," in *Proc. Int. Conf. Commun. Electron. Syst. (ICCES)*, Jul. 2019, pp. 409–416, doi: 10.1109/ICCES45898.2019.9002221.

[62] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, p. e4150, Jan. 2021, doi: 10.1002/ett.4150.

[63] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: Analysis on UNSW-NB15 data set and the real time online dataset," *Cluster Comput.*, vol. 23, no. 2, pp. 1397–1418, Jun. 2020, doi: 10.1007/s10586-019-03008-x.

[64] S. Krishnaveni, S. Sivamohan, S. S. Sridhar, and S. Prabakaran, "Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing," *Cluster Comput.*, vol. 24, no. 3, pp. 1761–1779, Sep. 2021, doi: 10.1007/s10586-020-03222-y.

[65] 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," *Inf. Secur. J., Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Apr. 2016, doi: 10.1080/19393555.2015.1125974.

[66] S. M. H. Bamakan, H. Wang, and Y. Shi, "Ramp loss K-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem," *Knowl.-Based Syst.*, vol. 126, pp. 113–126, Jun. 2017, doi: 10.1016/j.knosys.2017.03.012.

[67] N. Moustafa, G. Creech, E. Sitnikova, and M. Keshk, "Collaborative anomaly detection framework for handling big data of cloud computing," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2017, pp. 1–6, doi: 10.1109/MilCIS.2017.8190421.

[68] T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," in *Proc. IEEE 26th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2017, pp. 1881–1886, doi: 10.1109/ISIE.2017.8001537.

[69] P. Mishra, E. S. Pilli, V. Varadharajan, and U. Tupakula, "Out-VM monitoring for malicious network packet detection in cloud," in *Proc. ISEA Asia Secur. Privacy (ISEASP)*, 2017, pp. 1–10, doi: 10.1109/ISEASP.2017.7976995.

[70] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 481–494, Dec. 2019, doi: 10.1109/TBDATA.2017.2715166.

[71] I. Benmessahel, K. Xie, and M. Chellal, "A new evolutionary neural networks based on intrusion detection systems using multiverse optimization," *Appl. Intell.*, vol. 48, no. 8, pp. 2315–2327, Aug. 2018, doi: 10.1007/S10489-017-1085-Y.

[72] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, ''Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark,'' *IEEE Access*, vol. 6, pp. 59657–59671, 2018, doi: 10.1109/ACCESS.2018.2875045.

[73] V. Hajisalem and S. Babaie, ''A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection,'' *Comput. Netw.*, vol. 136, pp. 37–50, May 2018, doi: 10.1016/j.comnet.2018.02.028.

[74] N. M. Khan, N. Madhav C, A. Negi, and I. S. Thaseen, ''Analysis on improving the performance of machine learning models using feature selection technique,'' in *Proc. Int. Conf. Intell. Syst. Design Appl.*, in Advances in Intelligent Systems and Computing, vol. 941, 2018, pp. 69–77, doi: 10.1007/978-3-030-16660-1_7.

[75] P. Dahiya and D. K. Srivastava, ''A comparative evolution of unsupervised techniques for effective network intrusion detection in Hadoop,'' in *Proc. Int. Conf. Adv. Comput. Data Sci.*, in Communications in Computer and Information Science, vol. 906, 2018, pp. 279–287, doi: 10.1007/978-981-13-1813-9_28.

[76] W. Zong, Y. W. Chow, and W. Susilo, ''A two-stage classifier approach for network intrusion detection,'' in *Proc. Int. Conf. Inf. Secur. Pract. Exper.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 11125, 2018, pp. 329–340, doi: 10.1007/978-3-319-99807-7_20.

[77] J. Gao, S. Chai, B. Zhang, and Y. Xia, ''Research on network intrusion detection based on incremental extreme learning machine and adaptive principal component analysis,'' *Energies*, vol. 12, no. 7, p. 1223, Mar. 2019, doi: 10.3390/en12071223.

[78] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, ''TSDL: A two-stage deep learning model for efficient network intrusion detection,'' *IEEE Access*, vol. 7, pp. 30373–30385, 2019, doi: 10.1109/ACCESS.2019.2899721.

[79] Y. Yang, K. Zheng, C. Wu, and Y. Yang, ''Improving the classification effectiveness of intrusion detection by using improved conditional variational AutoEncoder and deep neural network,'' *Sensors*, vol. 19, no. 11, p. 2528, Jun. 2019, doi: 10.3390/s19112528.

[80] B. A. Tama, M. Comuzzi, and K.-H. Rhee, ''TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system,'' *IEEE Access*, vol. 7, pp. 94497–94507, 2019, doi: 10.1109/ACCESS.2019.2928048.

[81] C. Khammassi and S. Krichen, ''A NSGA2-LR wrapper approach for feature selection in network intrusion detection,'' *Comput. Netw.*, vol. 172, May 2020, Art. no. 107183, doi: 10.1016/j.comnet.2020.107183.

[82] J. Zhang, Y. Ling, X. Fu, X. Yang, G. Xiong, and R. Zhang, ''Model of the intrusion detection system based on the integration of spatial–temporal features,'' *Comput. Secur.*, vol. 89, Feb. 2020, Art. no. 101681, doi: 10.1016/j.cose.2019.101681.

[83] S. M. Kasongo and Y. Sun, ''A deep learning method with wrapper based feature extraction for wireless intrusion detection system,'' *Comput. Secur.*, vol. 92, May 2020, Art. no. 101752, doi: 10.1016/j.cose.2020.101752.

[84] A. S. Almogren, ''Intrusion detection in edge-of-things computing,'' *J. Parallel Distrib. Comput.*, vol. 137, pp. 259–265, Mar. 2020. Accessed: Jan. 26, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S074373151930872X

[85] S. M. Kasongo and Y. Sun, ''Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset,'' *J. Big Data*, vol. 7, no. 1, pp. 1–20, Dec. 2020, doi: 10.1186/s40537-020-00379-6.

[86] I. S. Thaseen, J. S. Banu, K. Lavanya, M. R. Ghalib, and K. Abhishek, ''An integrated intrusion detection system using correlation-based attribute selection and artificial neural network,'' *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 2, p. e4014, Feb. 2021, doi: 10.1002/ett.4014.

[87] A. Nazir and R. A. Khan, ''A novel combinatorial optimization based feature selection method for network intrusion detection,'' *Comput. Secur.*, vol. 102, Mar. 2021, Art. no. 102164. Accessed: Jan. 27, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820304375

[88] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha, ''Towards effective network intrusion detection: From concept to creation on Azure cloud,'' *IEEE Access*, vol. 9, pp. 19723–19742, 2021, doi: 10.1109/ACCESS.2021.3054688.

[89] H. Zhang, J.-L. Li, X.-M. Liu, and C. Dong, ''Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection,'' *Future Gener. Comput. Syst.*, vol. 122, pp. 130–143, Sep. 2021, doi: 10.1016/j.future.2021.03.024.

[90] J. Kaur, A. Agrawal, and R. A. Khan, ''P2ADF: A privacy-preserving attack detection framework in fog-IoT environment,'' *Int. J. Inf. Secur.*, vol. 22, no. 4, pp. 749–762, Aug. 2023, doi: 10.1007/s10207-023-00661-7.

[91] M. V. O. de Assis, L. F. Carvalho, J. J. P. C. Rodrigues, J. Lloret, and M. L. Proença Jr., ''Near real-time security system applied to SDN environments in IoT networks using convolutional neural network,'' *Comput. Electr. Eng.*, vol. 86, Sep. 2020, Art. no. 106738, doi: 10.1016/j.compeleceng.2020.106738.

[92] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, ''Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment,'' *IEEE Access*, vol. 8, pp. 83765–83781, 2020, doi: 10.1109/ACCESS.2020.2992044.

[93] J. P. A. Maranhão, J. P. C. L. da Costa, E. P. de Freitas, E. Javidi, and R. T. de Sousa Júnior, ''Error-robust distributed denial of service attack detection based on an average common feature extraction technique,'' *Sensors*, vol. 20, no. 20, pp. 1–21, Oct. 2020, doi: 10.3390/s20205845.

[94] D. C. Can, H. Q. Le, and Q. T. Ha, ''Detection of distributed denial of service attacks using automatic feature selection with enhancement for imbalance dataset,'' in *Proc. Asian Conf. Intell. Inf. Database Syst.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 12672, 2021, pp. 386–398, doi: 10.1007/978-3-030-73280-6_31.

[95] D. Akgun, S. Hizal, and U. Cavusoglu, ''A new DDoS attacks intrusion detection model based on deep learning for cybersecurity,'' *Comput. Secur.*, vol. 118, Jul. 2022, Art. no. 102748, doi: 10.1016/j.cose.2022.102748.

[96] R. K. Batchu and H. Seetha, ''An integrated approach explaining the detection of distributed denial of service attacks,'' *Comput. Netw.*, vol. 216, Oct. 2022, Art. no. 109269, doi: 10.1016/j.comnet.2022.109269.

[97] M. Almiani, A. Abughazleh, Y. Jararweh, and A. Razaque, ''Resilient back propagation neural network security model for containerized cloud computing,'' *Simul. Model. Pract. Theory*, vol. 118, Jul. 2022, Art. no. 102544, doi: 10.1016/j.simpat.2022.102544.

[98] H. Aydın, Z. Orman, and M. A. Aydın, ''A long short-term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment,'' *Comput. Secur.*, vol. 118, Jul. 2022, Art. no. 102725, doi: 10.1016/j.cose.2022.102725.

[99] A. Maheshwari, B. Mehraj, M. S. Khan, and M. S. Idrisi, ''An optimized weighted voting based ensemble model for DDoS attack detection and mitigation in SDN environment,'' *Microprocessors Microsystems*, vol. 89, Mar. 2022, Art. no. 104412, doi: 10.1016/j.micpro.2021.104412.

[100] L. D'hooge, M. Verkerken, T. Wauters, F. De Turck, and B. Volckaert, ''Castles built on sand: Observations from classifying academic cybersecurity datasets with minimalist methods,'' in *Proc. 8th Int. Conf. Internet Things, Big Data Secur.*, Apr. 2023, pp. 61–72, doi: 10.5220/0011853300003482.

MHAMAD BAKRO received the B.Sc. degree in informatics engineering from the Faculty of Informatics Engineering, University of Aleppo, Aleppo, Syria, in 2018 (with a final grade of 82.50%), and the M.Tech. degree (Hons.) in computer science and engineering from C. V. Raman Global University, Bhubaneswar, India, in 2020 (with a final grade of 85.10%). Since October 2018, he has been a full-time Researcher with C.V. Raman Global University. He has more than five years of teaching experience with C. V. Raman Global University. He has published three conferences, one book chapter, and one journal article. His research interests include AI for cybersecurity, cloud computing, intrusion detection systems, and optimization algorithms. He acts as a reviewer in many reputable journals and conferences.

RAKESH RANJAN KUMAR received the M.Tech. degree from MNNIT, Allahabad, India, and the Ph.D. degree from IIT (ISM), Dhanbad, India. He is currently an Assistant Professor with the Department of Computer Science and Engineering, C. V. Raman Global University, India. He has published more than 25 papers in reputed journals and conferences. His current research interests include cloud computing, service selection, and optimization. He acted as a reviewer in many reputed journals and conferences.

MOHAMMAD HUSAIN received the B.E. degree in computer science and engineering, in 1990, the M.Tech. degree in computer science, in 2005, and the Ph.D. degree in computer science and engineering, in 2008. He is currently with the Department of Computer Science, Faculty of Computer and Information Systems, Islamic University of Madinah, Saudi Arabia. He has about 33 years of professional experience in the field of IT and academics. He has industrial exposure and has experience in software design and development. He has published about 159 papers in different national/international journals/proceedings. He guided 14 Ph.D. and 11 M.Tech. Students. He has authored a book on principles of programming languages and contributed a number of book chapters. His research interests include information retrieval, web mining, and big data. He is an active member of various national/international journals, committees, and societies.

ZUBAIR ASHRAF received the master's degree in computer science and application from Aligarh Muslim University, Aligarh, India, in 2013, and the Ph.D. degree in computer science from South Asian University, New Delhi, India, in 2020. He is currently an Assistant Professor with the Department of Computer Engineering and Applications, GLA University, Uttar Pradesh, India. His research interests include machine learning, evolutionary optimization, nature-inspired intelligent computation, deep learning, and fuzzy systems. He is an IEEE Young Professional and an active member of several societies, including the IEEE Computational Intelligence Society and EUSFLAT. He is also an Active Reviewer of journals, such as IEEE Transactions on Fuzzy Systems, *Soft Computing*, *Applied Soft Computing*, *Journal of Applied Mathematics*, and the *International Journal of Intelligent Systems*.

ARSHAD ALI (Senior Member, IEEE) received the B.Sc. degree in mathematics and statistics, in 2000, the master's degree in computer sciences, the M.Sc. degree in telecommunication technology from Aston University, Birmingham, U.K., in 2007, and the Ph.D. degree from Lancaster University, U.K., in 2012. In 2005, he moved to Birmingham, U.K., for further studies. From 2007 to 2009, he was with the Geotechnical Group, Department of Engineering, and the University of Cambridge, as a Research Assistant. In December 2014, he joined the Islamic University of Madinah as an Assistant Professor, then promoted as an Associate Professor, in July 2018, and later he was promoted as a Full Professor of information technology with the Faculty of Computer and Information Systems, in June 2023. He worked on the U.K.-NEES project and designed a communication system for live experimentation between U.K. Universities, such as Cambridge University, Oxford University, and Bristol University. From 2009 to 2012, he was awarded a scholarship for the Ph.D. degree.

SYED IRFAN YAQOOB received the Ph.D. degree in computer science. He is currently an Assistant Professor and the Head of the BCA Program, School of Computer Science, Dr. Vishwanath Karad MIT World Peace University, Pune, India. He is also an accomplished academic and researcher in computer science. With years of experience, he has a strong track record of teaching, research, and development in the areas of cloud computing and software engineering. He is also well-versed in AI, machine learning, cryptography, and data science. His impressive educational background reflecting his commitment to academic excellence and research. Overall, he is a highly qualified and skilled professional in the field of computer science.

MOHAMMAD NADEEM AHMED is currently an Assistant Professor with the Department of Computer Science, King Khalid University, Abha, Saudi Arabia. He has more than seven years of experience in teaching and industry. He is cloud certified, java certificate, oracle developer certified, and big data certified. He has published several papers in Scopus, SCI-indexed journals, and conferences. His research interests include big data, machine learning, web security, object-oriented programming, and cloud computing.

NIKHAT PARVEEN received the B.Sc. degree in computer science and the M.C.A degree from Andhra University, Andhra Pradesh, India, in 2000 and 2003, respectively, and the Ph.D. degree from the Department of Computer Application, Integral University, Lucknow, Uttar Pradesh, India. She is currently an Associate Professor with the Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur, India. She has more than 12 years of teaching experience and six years of research experience. She has six national patents. Her research and publication interests include artificial intelligence, machine learning, security software, security testing, software engineering, and requirement engineering. She is also working in the areas of soft computing, image analysis, big data analytics, and the IoT. Her research has been chronicled in more than 30 journal publications and international conferences. She is a Life Time Member of CSI, ACM, IAENG, and IACSIT.