Machine Learning based Network Intrusion Detection Optimization for Cloud Computing Environments

Jitendra Kumar Samriya, Surendra Kumar, Mohit Kumar, Huaming Wu, *Senior Member, IEEE,* and Sukhpal Singh Gill

***Abstract*—Cloud computing is an emerging choice among businesses all over the world since it provides flexible and world wide web computer capabilities as a customizable service. Because of the dispersed nature of cloud services, security is a major problem. Since it is extremely accessible to intruders for any kind of assault, privacy and security are major hurdles to the on-demand service's success. A massive increase in network traffic has opened the path for increasingly difficult and broad security vulnerabilities. The use of traditional Intrusion Detection Systems (IDS) to prevent these attempts has proven ineffective. Therefore, this paper proposes a novel Network Intrusion Detection System (NIDS) based on a Machine Learning (ML) model known as the Support Vector Machine (SVM) and eXtreme Gradient Boosting (XGBoost) techniques. Furthermore, the hyperparameter optimization technique based on the Crow Search Algorithm is being utilized to optimize the NIDS' performance. Besides, the XGBoost-based feature selection technique is used to improve the classification accuracy of NIDS's method. Finally, the performance of the proposed system is evaluated using the NSL-KDD and UNR-IDD datasets, and the experiment results show that it performs better than baselines and has the potential to be used in modern NIDS.**

***Index Terms*—Cloud Computing, Machine Learning, Network Intrusion Detection, Performance Optimization.**

I. Introduction

IN view of the rising prevalence of the Internet in modern society, there has been a greater need for items that can connect to networks, and as a result, the issue of privacy and security has gained more prominence [1], [2]. The trust issue in shared virtualization technology deployed in the cloud network is the main obstacle in developing secured decentralized applications [3]. The majority of reported intrusion cases have recently risen, exposing severe threats to organizations, businesses, and people [4]. Distributed Denial of Service (DDoS) and ransomware are some of the malicious cyber-attacks that affect cloud-based networks [5], [6]. The sophistication and complexity of networks are increased by these attacks [7]. Due to this, data confidentiality, availability and integrity are compromised. It is crucial to be able to detect and respond to these kinds of threats that implement essential mitigation as well as restrict any harm that affects the cloud network [8]. Traditional approaches for network security such as antivirus, firewall and access control are insufficient to recognize new intrusions. So it is important to adopt the next stage of protection. Thus, an Intrusion Detection System (IDS) is known as the promising solution to detect both known attacks as well as unknown threats [9]. It is developed to ensure system protection and can identify abnormalities quickly. Moreover, it can also increase device stability and protection by reacting and detecting a variety of malicious behaviors [10], [11]. The IDS may be categorized as Hybrid systems, Host-based or Network-based. Host-based IDS (HIDS) examines and tracks internal device data for unauthorized behavior, such as key system files, program logs, and operating system audit reports [12]. Network-based IDS (NIDS) tracks the network traffic for doubtful activities and is deployed at multiple points or strategic points in the network [13]. In the hybrid approach, the IDS can identify the attacks from network or host sources [14]. Even if anomaly detection systems are more effective at identifying unknown attacks, they often generate a high-level false positive rate.

A misuse detection system can address this limitation, which depends on particular attack patterns to differentiate malicious activities from normal activities [15]. However, the detection rules influence these systems directly. Therefore, the learning speed of IDS and improving the detection accuracy is a difficult task [16]. To overcome this, a Support Vector Machine (SVM) based Network Intrusion Detection System (NIDS) is proposed to detect malicious threats with high accuracy, F-Score, Precision and Recall. Compared to other learning methods such as Artificial Neural Networks (ANN), Multi-Layer Perception and Naive Bayes, SVM is an efficient classification approach [17]. It is based on statistical learning theory, which is commonly used to solve classification, pattern and voice recognition problems. SVM can overcome the challenges of high dimensionality, nonlinearity and small samples better than other classification approaches [18]. Hyperplanes are used by the SVM classifier to distinguish various class labels. SVM with kernel functions is used to solve the nonlinear classification problems [19].

Unidentified network traffic information can be gathered

J. K. Samriya is with the Department of CSE, IIIT Sonepat, India. Email: jitu.samriya@gmail.com.

S. Kumar is with the Department of Computer Engineering and Applications, GLA University, Mathura, INDIA. Email: kumar.surendra1989@gmail.com.

M. Kumar is with the Department of IT, Dr. B.R. Ambedkar National Institute of Technology, Jalandhar, INDIA. Email: kumarmohit@nitj.ac.in.

H. Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin, China. Email: whming@tju.edu.cn

S. S. Gill is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, United Kingdom. Email: s.s.gill@qmul.ac.uk.

(Corresponding author: Huaming Wu)

from various network sources, and machine learning techniques can be employed to produce a high-quality description of features from datasets. These characteristics can be utilized for classification on a small, labeled traffic dataset that includes both regular and abnormal traffic records. The traffic data for the dataset with labels can be gathered within a restricted, segregated, and secure network environment. Present methods cannot be depended upon to maintain the required levels of accuracy. Thus, it is necessary to have higher levels of specificity, thoroughness, and comprehension in order to have a more holistic and precise perspective. However, this carries a range of financial, computational, and time-based expenses.

To achieve high accuracy, the intrusion detection approach can be hybrid with nature-inspired algorithms such as harmony search algorithm (HSA), simulated annealing (SA), particle swarm optimization (PSO), Artificial Bee Colony algorithm (ABC), Spider Monkey Optimization (SMO), etc. Because these algorithms have the capability to decrease execution time and improve prediction accuracy. Therefore, to optimize the SVM parameters, the Crow Search Algorithm (CSA) is used in the proposed approach called **CSA-SVM**.

The following are major contributions of this research work:

- Determine the best set of features in the NSL-KDD and UNR-IDD dataset, which consists of both normal data and the most recent and frequent attacks. The purpose of this study is to utilise the XGBoost algorithm and identify a specific set of features that would enhance the performance of the detection mechanism.
- Optimize the Kernel parameters of the SVM classifier by employing the Crow Search Algorithm to maximise the rate of detecting intrusions.
- Evaluate the performance to test the proposed approach using two network datasets named NSL-KDD and UNR-IDD. The experimental results are employed to determine its effectiveness as compared to state-of-the-art approaches.

The remainder of the paper is organized as follows: In Section II, the details of earlier approaches are illustrated and deal with a problem statement. In Section III, the complete details of the proposed model are illustrated. In Section IV, the experimental setup is explained in detail. Section V deals with the analysis of the results. Finally, Section VI concludes the paper.

II. Related Work

IDS is among the cloud security industry's most challenging areas. Viruses, worms, password cracking, denial of service, scanning, and malware code insertion are all becoming more widespread in the cloud. If not identified promptly, these assaults endanger the company's image and result in financial damage. Many scholars have presented numerous approaches to identify an attack in cloud environments in the past. The Oppositional Crow Search Algorithm (OCSA) based attack detection was conducted by [20]. To detect a DoS attack, Opposition Based Learning (OBL) and Crow Search Algorithm (CSA) were integrated. This approach includes two sections. In the first section, OCSA-based feature selection was conducted. In the second stage, the classification process was accompanied by the use of a Recurrent Neural Network (RNN) classifier. The authors used a benchmark dataset to validate the suggested technique. To detect DoS attacks, [21] suggested a rule-based classification system in cloud networks. The ranking and scoring algorithms were used for feature selection. Then rule-based classification algorithm was implemented as a classifier which is based on expert knowledge. For performance evaluation, 5000 attack instances were selected and they are split into 5 datasets. The existing algorithms named Naive Bayes, Multilayer perceptron, SVM, and Decision tree were used for performance comparison. The combination of various machine learning approaches was suggested by [22] to develop the hybrid IDS. Initially, the normalization and transformation were executed. Then, the classification with different machine learning approaches was conducted. The performance evaluation was carried out based on the Matthews Correlation Coefficient (MCC), False Positive Rate, True Positive Rate, F-measure and accuracy. For intrusion detection, two approaches were combined by [23]. It dealt with the packet scrutinization algorithm and Normalized K-means clustering algorithm with RNN (NK-RNN) approach. To examine the packets, a packet scrutinization algorithm was used while the NK-RNN was used for Classification. Moreover, for the benefit of cloud users who need to access cloud data a one-time signature approach was suggested to protect themselves from hackers. A combination of SVM with a fuzzy c means clustering (FCM) was presented by [24] for all types of attacks. For performance assessment, the NSL-KDD dataset and some state-of-art techniques were used with the performance metrics named F1-score, Recall, Precision, True Positive (TP) rate, False Negative (FN) rate, Incorrect Classification rate and Accuracy. Taj *et al.* [25] developed an IDS that integrates machine learning and data mining techniques. Initially, they conducted the data preparation and pre-processing. Then J48 Classifier, OneR Classifier, Naïve Bayes and Hoeffding Tree were used as the classifiers. To classify data, a Pre-classified dataset was used. Finally, the author concluded that, the J48 classifier has outperformed better efficiency than the other classifiers. Devan *et al.* [26] have presented a DNN-based classification approach for IDS. Initially, they performed normalization and then the XGBoost technique was implemented for feature selection. For the classification process, Deep Neural Network (DNN) was employed. During the training process of DNN, the Adam optimizer is used to maximize the learning rate. For performance assessment, some state-of-the-art techniques were utilized. To detect intrusion, Hajimirzaei *et al.* [27] suggested a technique based on Artificial Bee Colony (ABC), Fuzzy clustering algorithms, and Multi-Layer Perceptron (MLP) network. To create training subsets, a fuzzy clustering approach was used. ABC algorithm was used for network training. It optimizes the bias and linkage weights of the MLP. Finally, the attacks were classified by the MLP network. For performance evaluation, Mean absolute error (MAE), kappa statistic, and root mean square error (RMSE) were utilized. Tummalapalli *et al.* [28] presented the Gravitational Group Search-based Support Vector Neural Network (GG-SVNN) classifier for attack detection. For clustering the nodes, the Bayesian fuzzy

TABLE I: Comparison of Existing Approaches

Year	Type of IDS	Techniques	Advantages	Disadvantages
2021 [20]	Network-based ID	OCSA-based feature selection and RNN-based classification	In terms of speed and memory requirements, this method is low-cost.	Result not applicable for all classes. It detects only DoS attack
2019 [21]	Network-based ID	Scoring and Ranking-based Feature Selection and Rule-based Classification Algorithm	High security and a lower false alarm rate	Because of the dynamic nature of threats, unpredictability was not managed. Moreover, the Rule creation process is time-consuming.
2019 [22]	Network-based ID	The CfsSubsetEval algorithm-based feature selection and compared the performance of various machine learning algorithms	Based on the attack type, this approach selects the appropriate machine learning algorithm for classification. Hence it provides high accuracy.	Computationally expensive because analyzing the performance of various machine learning algorithms and selecting the appropriate algorithm for the particular attack is time-consuming.
2019 [23]	NID-based	Packet Routinization algorithm based feature selection and NK-RNN based classification	It can identify both low and high-frequency attacks	Leads to higher loss ratio when there is an increase in Packets
2020 [24]	Network based ID	Hybrid FCM-SVM for classification	It quickly detects new incoming traffic.	Large datasets were used but the feature selection techniques were not used. Hence, the processing time may increase.
2020 [25]	HIDS	Combination of data mining and machine learning algorithms	self-created dataset is used to find more attacks	The J48 classifier have higher accuracy but with increased runtime complexity.
2020 [26]	Network-based ID	XG-Boost based feature selection and DNN-based classification	Regularization is dealt with by XGBoost, and it provides the prevention of overfitting problems.	It's difficult to train several secret layers in DNN. Parallelism in neurons requires multi-core platforms, GPUs, and CPUs.
2019 [27]	Network-based ID	MLP is used for classification, ABC algorithm is used to update the weight of MLP and fuzzy is used to create the training subset	This IDS model is very strong	The structure of the fuzzy and ABC algorithms is extremely complex. It causes increased training and testing time.
2020 [28]	Network-based ID	To clustering the node, Bayesian fuzzy clustering is used then GG-SVNN based classification is performed	It classifies the attacks very effectively by the use of the clustering approach	Large datasets were used but the feature selection techniques are not used. As a consequence of this condition, processing times are extended.
2020 [29]	Network-based ID	V-ELM based intrusion detection	The use of multiple ELMs improves detection accuracy while lowering false alarms.	Due to the incorporation of a voting scheme, the training time is increased.
2020 [30]	Network-based ID	HKELM-based classification and DEGSA-based parameter selection	Testing and training time are very less	Its performance depends on some input parameters.
2020 [31]	Network-based ID	PFCM-based clustering and GSA-FIS-based classification	It achieves better detection rates	Computational cost is very high due to the clustering approach.
2020 [32]	Network-based ID	Improved deep belief network	(DBN) for classification and CD algorithm is used for training	Overfitting problem is reduced The output of DBN is not satisfactory because the input information is clamped due to the CD algorithm (pre-training algorithm)
2019 [33]	Network-based ID	SVM-based classification and QPSO are used to optimize the SVM parameters	Lower classification error rates across a variety of datasets	Long computational time and complexity because feature selection is not done.

clustering technique was utilized. For classification, the authors combine the Gravitational Search Algorithm (GSA) and the group search optimizer (GSO). They used the KDD cup dataset to evaluate the suggested approach. For performance comparison, K-means clustering with Neural Networks (NN), Support Vector Neural Networks (SVNN) with Fuzzy c-means (FCM) and Fuzzy Clustering-based Neural Networks were implemented. A Machine Learning (ML)-based system has been developed in [29] to monitor DDoS attacks in a cloud computing environment. To design the system Vulnerability Enhanced Learning Machine (VELM) was used as a classifier. The same training data set was used for all Extreme Learning Machines (ELMs) in the VELM for training. The output is calculated and it is applied to the sample of each ELM during the detection of attack. At last, the majority voting was taken by using the samples. Two benchmark datasets were used to evaluate the performance. A Hybrid Kernel Extreme Learning Machine (HKELM) model based on the hybrid kernel function was presented [30] to detect various attacks accurately. To enhance the parameters of HKELM, the combination of the Differential Evolution (DE) algorithm and GSA was applied that improve the global optimization during prediction attacks. For feature extraction and dimensionality reduction, the Kernel Principal Component Analysis was implemented. Thus, a combination of Kernel Principal Component Analysis (KPCA), HKELM and DE-GSA was obtained for anomaly detection. Shyla *et al.* [31] have presented a gravitational search algorithm-based Fuzzy Inference System (FIS) for detecting the attacks. To enhance the parameters of Fuzzy, GSA was introduced. The received packets were pre-processed and clustered by Possibilistic FCM (PFCM) which monitors the noise of FCM. The clustered packets were provided to the tuneable fuzzy inference framework after the clustering step. The fuzzy score monitors the usual and anomalous packets. GSA performs the training by improving the whole fuzzy framework. As a result, four kinds of abnormal data are identified: DoS, Remote to Local (R2L), User to Root (U2R) and Probe. The process of data pre-processing Probability Mass Function (PMF) and Min-Max methods was implemented for simplification in Deep Belief Networks (DBNs) [32]. Additionally, non-mean Gaussian distribution and Kullback-Leibler (KL) divergence, based on a combined sparsity penalty, were implemented into the learning phase function of the model. - During this implementation, the sparse constraint is obtained from sparse distribution, which induced the sparse layer of the hidden layer neurons. This will avoid feature homogeneity and network overfitting. On comparing with other IDS techniques the performance of false positive rate and accuracy performs better results. Quantum-Behaved Particle Swarm Optimization and SVM (QPSO-SVM) model was introduced in [33] for intrusion detection to enhance the parameters of SVM. It enhanced the outcomes of their framework. In linear and nonlinear classification problems, this method changes the parameter values to obtain the best-separating hyperplane. From the analysis of the literature review, it is observed that the previously proposed models have their own disadvantages and advantages which are listed below in Table I.

Classification methods in machine learning models take a long time because of high-dimensional features. Since the dataset used in this study has a large number of dimensions, an efficient dimensionality reduction mechanism was required to reduce the classifier's workload. Furthermore, a features selection mechanism will assist the classifier to choose the most important attributes and remove those that have a negative

effect on the efficiency of the classifier. This prompted us to generate a framework capable of selecting the most appropriate attributes from the IDS dataset. In addition to that, an excessive volume of data leads to an increase in false alarm reports of intrusion and reduces the accuracy of detection. This is one of the major issues when the system encounters unknown attacks. We have used machine learning algorithms like XGBoost and SVM to meet the aforementioned challenges. The proposed model's efficiency is then analyzed by comparing it with other techniques.

III. Methodology

Cloud computing is a platform that allows users to share resources, services, and information. It gives companies a flexible framework that makes computing more efficient. The adoption of cloud computing environments by a wide range of organizations has brought with it a slew of issues and challenges. They use servers as a service on the Internet, so issues like user privacy, data leakage, and authentication are still major concerns in the cloud environment [27]. The cloud environment is a collection of resources for providing cloud users with on-demand administrations. The web provides access to the cloud environment, making data stored on the cloud more accessible to both intrinsic and extrinsic attackers. Since every typical client works in the cloud, there's a high possibility that an attack will occur. To identify data that has been compromised, several IDS have been designed [34].

The IDS is critical to the network's security. IDS, in general, collects incoming data across the network. These data are transferred to a pre-processing system, which filters out noise and substitutes attributes that are unnecessary or misconstrued [35], [36]. The data that is pre-processed are analyzed and classified according to severity. If it is a normal record, no additional adjustments are necessary; if not, it is directed to reporting generation, which generates an alert. The complexity of the issue is used to trigger alerts. False negative, false positive, true negative and true positive detection reports are provided by the IDS system. True positive outcomes occur when an actual threat is detected and the IDS responds by generating a warning. When no attack occurs and the IDS does not generate the alert, this is referred to as a true negative [37]. False positive, also known as false alarm, is a major flaw in the system that happens when an IDS detects no attack. When an IDS misses a prospective or true attack, it is called a false negative. It leads to false-negative rates or low detection rates, if these technologies don't achieve the expected results [38], [39]. This issue can be solved through an effective attack detection system for cloud infrastructure, which is the focus of our research.

Our proposed system consists of three modules, namely, (i) pre-processing (ii) XGBoost for feature selection and (iii) intrusion detection using SVM classifier. Initially, we pre-process the features for further processing. After pre-processing, the normalization method is used to convert the values of numeric columns in a dataset to a similar scale without distorting variations in value ranges. Then we select the important features using the XGBoost algorithm to increase in accuracy of classification. The whole process is split into two stages, such as testing and training. Later, selected features are classified by a multi-class SVM classifier. Adding to this, optimization of classifier parameters is done by the Crow Search Algorithm. Fig. 1 displays the overall idea of the proposed framework. Pre-processing, Feature Selection and Classifier are the three steps to be followed.

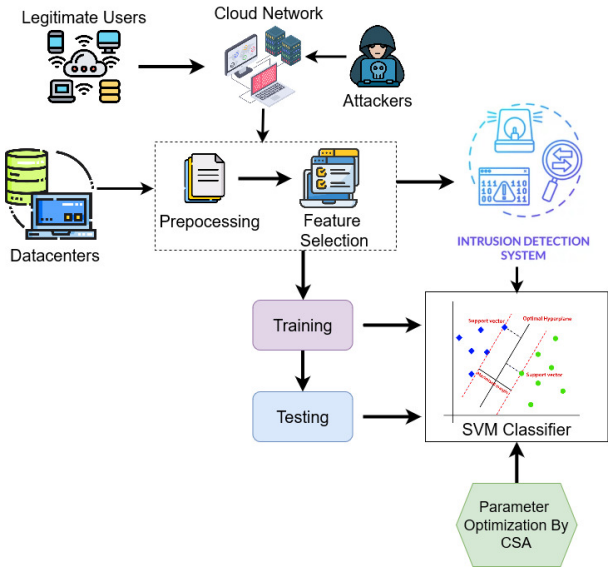

Fig. 1: Proposed System Architecture for Intrusion Detection

Here, we use the NSL-KDD (Network Security Laboratory - Knowledge Discovery in Databases) and UNR-IDD (University of Neveda-Reno Intrusion Detection) datasets. From the NSL-KDD dataset, 41 features are classified as normal or specific attack types and are categorized into four groups, namely, host-based traffic features, time-based traffic features, content features and basic features. Feature values in the dataset can be in the form of Symbolic, Discrete and Continuous values. The UNR-IDD dataset has 29 features. The characteristics of the datasets can be categorized as follows,

- Without inspecting the packet, basic features may be derived from a TCP/IP channel.
- The features in the same Host Traffic are evaluated in a 2-second window, and only when the current connection is the same as that of the host.
- In the Same Service Traffic Features, they are evaluated in a 2-second window, and only those connections with the same service are considered.
- Content features are used to trace the suspicious activity.

A. Data Pre-Processing

The dataset has undergone two operations such as transformation and normalization. This process is used to remove redundant data and produces better output results. So that a more optimized dataset is extracted in this case.

B. Transformation

In the dataset, the symbolic attributes are transformed into numerical ones. These attributes in the form of Service, flag and protocol in the dataset are converted. Thus the dataset is

completely composed of numerical and is further processed for classification operations. Protocols (udp changed to 1, icmp to 2 and tcp to 3) flag and service attributes have been converted to numerical values.

C. Normalization

Normalization of datasets is an essential pre-processing technique. It is used for dataset training and testing. Min-Max and Gaussian are the most widely known Normalization methods. In this case, we have used the Min-Max normalization approach. The minimum and maximum values in a group are used in this process. Additional data are normalized to these values. The main goal of this is to set the data with a minimum value of 0 and a maximum value of 1. The formula for calculating the new value is as follows:

$$D' = \frac{D - D_{min}}{D_{max} - D_{min}} X[new_{min} - new_{max}] + new_{min}. \quad (1)$$

D. Feature Selection Using XGBoost Model

The main intention of the selection of features is to find the best features in the data set. Data can be classified by machine learning algorithms into a collection of class features and goals. To overcome the problem of reducing irrelevant and unnecessary variables, various techniques have been developed. Feature selection (variable elimination) reduces computing needs, reduces dimensional curse effects, helps understand the data and improves the performance.

This paper focuses on the XGBoost technique for feature selection. The enhancing technique Xtreme gradient boosting (XGBoost) is a part of the ensemble-based method. The XGBoost algorithm has been known as a useful method for optimizing the gradient boosting algorithm by avoiding overfitting issues and eliminating missing values by parallel processing. Because of its performance and scalability, XGBoost is getting popular. Using gradient-boosted decision trees, it was designed specifically for speed and reliability. The technique is very effective in terms of computing time and memory use. It can handle missing values or be Sparse Aware, enhances parallel tree construction, and has the unique ability to boost data that has already been added to the trained model (Continued Training). XGBoost model has been widely known for its performance in various machine learning and data mining challenges. Thus, the XGBoost feature importance score technique is employed for selecting features. The prediction is done by the set of decision trees,

$$y_i' = \sum_{m=1}^{m} f_k(x_i), \quad (2)$$

where prediction from a decision tree as y_i', feature vector x at the i^{th} data point is denoted as f_k, and the number of decision trees is denoted as m. For training the model, a loss function needs to be optimized. For binary classification, the loss function is as follows,

$$L = -\frac{1}{N} \sum_{i}^{N} \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right]. \quad (3)$$

Regularization is a significant component of the XGBoost framework and is formulated as:

$$\Omega = \gamma^T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_{j^2}, \quad (4)$$

where the score of j^{th} leaf is denoted by w_{j^2} and the number of leaves is denoted by T. The object of the model is specified as follows,

$$obj = L + \Omega. \quad (5)$$

To optimize the objective function in XGBoost, the gradient descent order employs the variance and mean, which is expressed as:

$$obj(t) = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t). \quad (6)$$

To test the accuracy of the XGBoost Algorithm, feature selection is important. For this purpose future important score is used experimented by multiple thresholds. The function importance score basically permits evaluating every subset of importance features for each input variable. The test begins with the entire features and ends up with a subset of the most relevant ones. The selected features are further trained by the SVM Algorithm. The basic structure of the XGBoost algorithm is shown in Fig. 2.

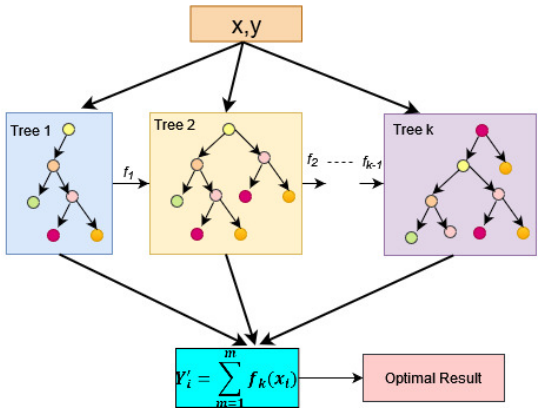

Fig. 2: Structure of XGBoost

Algorithm 1 Feature Selection

Require: Input: dataset.

1: Pre-processing.
2: Normalization using the min-max method in Eq. 1.
3: For selecting features, the XGBoost feature importance score is implemented.
4: Design Support Vector classifier using the feature set with XGBoost chosen in Step 3.
5: Dataset Training and Testing.
6: Design SVM Model: $SVMClassification = Train\_SVM(Normalized\_data, Feature\_Set)$.

E. SVM Classification Mechanism

The SVM framework can be simplified as a method for determining the best hyperplane which is used as the two-class

separator. In SVM, discrimination borders or the additional line may be reduced to two class members, +1 and 1. A margin is specified as the shortest distance between the hyperplane and the nearest pattern of each class. The closest pattern is used to generate the support vector. Thus, the identification of the best hyperplane position is said to be at the core of the SVM algorithm. They're common in security software like network anomaly detection. Every data point in n-dimensional space is plotted in this algorithm. After determining the value of a specific coordinate, classification is carried out by locating the hyperplane that best separates the classes. Two parameters, Cost and Gamma, are essential in classification. Gamma is a parameter that controls the impact of training examples on the newly created model. Moreover, the Cost parameter determines the cost of misclassification on training examples.

The aim of ε-SVM classification is to find a function $f(t)$ from the attained y_i target for the entire training data that has at most ε deviation. The ε-insensitive loss function is:

$$we\ choose \begin{cases} \text{Normal}, & \text{if } f(x) \geq 0, \\ \text{Anomaly}, & \text{otherwise.} \end{cases} \tag{7}$$

$$e(f(t)-y) = \begin{cases} 0, & \text{if } |f(t)-y| \leq \varepsilon, \\ |f(t)-y|-\varepsilon, & \text{otherwise.} \end{cases} \tag{8}$$

Simply by mapping the training data $\phi : t \in R^n \rightarrow F$ into feature space F of high dimension, Nonlinear SVM regression is obtained. The following expression for $f(t)$ is

$$f(t) = w \cdot t = b. \tag{9}$$

In feature space F, the best fitting function is estimated as follows:

$$f(t) = w' \cdot \phi(t) + b. \tag{10}$$

The term "flatness" refers to the ability to seek out. The issue is formalized by convex optimization issue by the following equation:

$$\min \ \frac{1}{2}||w||^2 + C\sum_{i=1}^{P}(\xi_i + \xi_i^*), \tag{11}$$

$$s.t. \ y_i - (w'\phi(t_i)+b) \leq \varepsilon - \xi_i, \tag{12}$$

$$(w'\phi(t_i)+b) - y_i \leq \varepsilon - \xi_i, \tag{13}$$

$$\xi_i, \xi_i^* \geq 0, i = 1, 2, \cdots, p; C > 0, \tag{14}$$

where the penalty factor C is regularization consistent, ξ_i^* and ξ_i denotes slack factors, ε indicates the tube size of SVM. User assesses ε and C based on observation. The trade-off between the flatness function and the number of deviations greater than ε endured is determined by C. The sorted capacity has the following format during the optimal solution,

$$\min \ -\frac{1}{2}\sum_{i,j=1}^{P}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(x_i.x_j) + \sum_{i=1}^{P} y_i(\alpha_i - \alpha_i^*) - \varepsilon\sum_{i=1}^{P}(\alpha_i + \alpha_i^*), \tag{15}$$

$$s.t. \ \sum_{i=1}^{P}(\alpha_i - \alpha_i^*) = 0, \tag{16}$$

$$0 \leq \alpha_i, \alpha_i^* \leq C, \tag{17}$$

$$f(t) = sgn\sum_{i,j=1}^{P}(\alpha_i - \alpha_i^*)K(t_i, t_j) + b, \tag{18}$$

where Lagrange multiplier coefficients are denoted by α_i and α_i^*, which are determined by solving the dual optimization problem in support vector learning. The training set, corresponding to $\alpha_i = \alpha_i^*$, is associated with the support vectors, where $K(t_i, t_j)$ denotes a kernel function. The optimization process is initiated using the Karush-Kuhn-Tucker (KKT) conditions. In this study, the Radial Basis Function (RBF) kernel function is implemented and expressed as:

$$K(t, t_i) = \phi(t)^T\phi(t_i) = \exp\left[\frac{-(||t-t_i||)^2}{\sigma^2}\right], \sigma.\varepsilon.R. \tag{19}$$

To achieve enhanced performance, many SVM parameters, including the regularization parameter *C* must be selected, the kernel parameter γ and Tube Size ε. In this research, CSA is proposed to optimize these three parameters.

F. Crow Search Algorithm

Crows are known to be the cleverest among birds as they have the widest brains in contrast to their body size. Based on the brain-to-body ratio, a crow's brain is slightly less. There are numerous examples of crows' brilliance. They can recognize each other's faces and alert each other when an intruder approaches. Moreover, they can connect in powerful ways and remember where their food was hidden for months. They keep an eye on various birds and follow them secretly to find where they store the food. They steal the food when the birds leave their location. When a crow performs stealing, it may take precautions, like shifting the hiding places to avoid becoming a suspect in the future. They use their intelligence of being such a thief to predict thief activity and determine the safest way to protect their food from being looted. The following are the principles of Crow Search Algorithm,

- They live in groups.
- They can recall the position of their hiding spots.
- They swam one up to another to steal.
- They guard their hideouts against theft by probability.

G. Mathematical Model for Crow Search Algorithm

Consider that there is an N-dimensional environment with several crows. C denotes the group size and the crow's position

i (iteration) at the time, $iter$ in the environment is expressed as:

$$N^{m,iter+1} = (p = 1, 2, \cdots, C; iter = 1, 2, \cdots, iter_{max}), \tag{20}$$

where $N^{m,iter} = \lfloor N_1^{m,iter}, N_2^{m,iter}, \cdots, N_C^{m,iter} \rfloor, iter_{max}$, considered as maximum count of iterations. The location of their hideouts was recorded in the memory of every crow. The position of the crow's secret place is defined as Suat iteration. This is considered to be the best position that Crow *u* has achieved. In each crow's memory, the better location that it achieved will be recorded. Till that it will look for a better position in the search space. During iteration *t*, let's assume crow j needs to go to its secret location. In the meantime, crow *u* tracks the secret location of crow *v* by following it. There are two possible events in this case:

Scenario 1: Crow *v* is unaware that he is being tracked by Crow *u*. As a matter of fact, crow *u* will reach the crow *v*s hiding food location, and the Crow *u*'s new location is updated as:

$$N^{m,iter+1} = N^{m,iter} + K_j X fll^{m,iter} X (S^{n,iter} - V^{m,iter}), \tag{21}$$

where randomization k_j has a uniform distribution between 0 and 1. During iteration flight length of crow *u* is denoted as $fll^{u,iter}$. It leads to local search when the value is less and results in global search when the value is high

Scenario 2: In this, Crow *v* is aware that Crow u is attempting to track it down. Thus, Crow *v* can trick Crow *u* by switching to a better random position in the environment. As a whole, scenarios 1 and 2 can be written in the following form:

$$N^{m,iter} = \begin{cases} N^{m,iter} + K_j X fll^{n,iter} X (S^{n,iter} - V^{m,iter}), \\ \qquad\qquad\qquad\qquad if\ K_j \geq AWP^{n,iter} \\ \text{a random location}, \qquad \text{otherwise} \end{cases} \tag{22}$$

where $AWP^{v,iter}$ denotes crow v's awareness probability during iteration.

Step 1: Parameter Initialization The decision variables, problem optimization and constraints are mentioned. Size of flock (N), flight length (fl), awareness probability (AP) and the maximum number of iterations ($iter_{max}$) are the adjustable parameters of CSA and are then evaluated.

Step 2: Initialization of crow's memory and location As flock members, N crows are randomly placed in a d-dimensional environment. Each crow denotes a feasible solution, and d_v stands for decision variables count.

$$Crows = \begin{bmatrix} n_1^1 & n_2^1 & \cdots & n_{dv}^1 \\ n_1^2 & n_2^2 & \cdots & n_{dv}^2 \\ \vdots & \vdots & \vdots & \vdots \\ n_1^c & n_2^c & \cdots & n_{dv}^c \end{bmatrix}. \tag{23}$$

Initialization of memory is done in every single Crow. Since the crows have no knowledge in the first iteration, it is considered that they have stored their foods in their original state.

$$Crows = \begin{bmatrix} m_1^1 & m_2^1 & \cdots & m_{dv}^1 \\ m_1^2 & m_2^2 & \cdots & m_{dv}^2 \\ \vdots & \vdots & \vdots & \vdots \\ m_1^c & m_2^c & \cdots & m_{dv}^c \end{bmatrix} \tag{24}$$

Step 3: Examine the fitness function The accuracy of the position of each crow is calculated by integrating the decision variable values into the fitness function. R_i represents the position of crow i in a population of N crows. δ^m and δ^n indicate the highest and lowest points of the $AWP^{v,iter}$, respectively.

$$AWP^{v,iter} = \delta^{min} + (\delta^{max} - \delta^{min}) \frac{R_i}{N}. \tag{25}$$

Step 4: Create new position As shown below, every crow creates a new place in the search space. Assume a Crow (say, *u*) wants to create a new location. For that, one crow (say, crow *v*) is randomly selected among the group and monitored to locate the hidden place of the food that the crow (S^v) has guarded. The Crow *u*'s new location is expressed. Each crow follows the same procedure.

Step 5: Examine the feasibility of new locations Examine all possibilities for every crow to find a novel position. If a crow's new location is achievable, then it updates the location. If not, they remain within their current location and will not switch to the newly achieved position.

Step 6: Evaluate new locations for fitness function The fitness function's value for each crow's novel location is assessed.

Step 7: Memory update of crow The updated crow's memory is evaluated as follows:

$$S^{u,iter} = \begin{cases} V^{u,iter}, \text{ if } f(V^{u,iter+1}) \geq f(S^{u,iter}), \\ S^{u,iter}, \quad \text{otherwise.} \end{cases} \tag{26}$$

where $f()$ denotes the value of the fitness function. It updates the memory of a new place when the value of the fitness function of a new place is better than the fitness function's value of the memorized location.

Step 8: Check the criteria for termination Steps 4-7 are iteratively executed until the maximum iteration ($iter_{max}$) is reached. The process continues until the best memory location, determined by the fitness function, is identified as the optimized solution.

IV. Experimental Setup

To evaluate the performance of the proposed approach, we employed an Intel (R) Xeon (R) system with the Processor E5-2640 (2.50 GHz) and 16 GB of primary memory for the experimental simulation. By using MATLAB Simulink tool[1] the performance is evaluated using the dataset named NSL-KDD[2] and UNR-IDD[3]. Our proposed work is compared with other Algorithms such as ANN, RNN and DNN. Fig. 3 shows our proposed method based on SVM-CSA.

[1]https://in.mathworks.com/products/simulink.html?s_tid=hp_ff_p_simulink
[2]https://www.kaggle.com/datasets/hassan06/nslkdd
[3]https://www.tapadhirdas.com/unr-idd-dataset

Algorithm 2 Proposed CSA-SVM Method

Require: Inputs for SVM: Normalised DataSet; for CRO: Initialize the value for group size (C), $iter_{max}$, *fll* and AWP

Determine the Solution's fitness values.
Set iteration *iter*
Set best search value
while $iter < iter_{max}$ **do**
  Update each data point and search the position using levy flight
  for $u = 1 : C$ (each crow of the group) **do**
    Pick any one crow to track (say,*v*)
    Set a probability of Awareness
    if $K_j \geq AWP^{n,iter}$ **then**
$V^{m,iter} = V^{m,iter} + K_j X fll^{n,iter} X(S^{n,iter} - V^{n,iter})$
    else
$N^{m,iter}$ = a random position of search space
    end if
    Check the feasibility of new places, the new location of the crows
    if new position > current position **then**
      Upgrade the memory of crows
    end if
    Assess the classifier's parameters.
    Proceed with step 6 when the recent objective value does not satisfy the termination condition. Otherwise, continue the process.
  Steps 6-15 are continued until finding the optimal solution
    Obtain the SVM model's optimal parameters C, ε and σ.
    This optimized SVM model is used as the intrusion detection process
  end for
end while

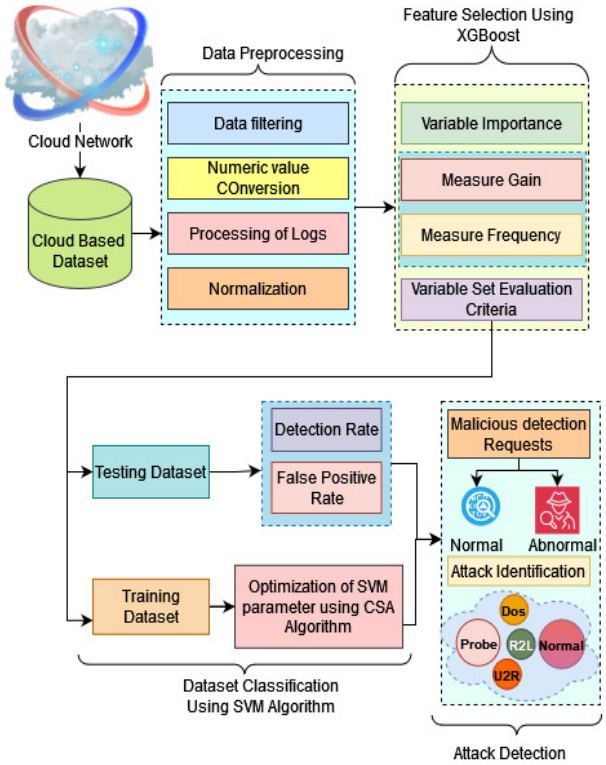

Fig. 3: The proposed SVM-CSA for Intrusion Detection

TABLE II: NSL-KDD and UNR-IDD features description

Dataset	Feature	Dataset	Feature
NSL-KDD	Protocol_Type Service Flag src_bytes dst_bytes wrong_fragment root_shell Count	NSL-KDD	srvs_error_rate same_srv_rate diff_srv_rate dst_host_diff_srv_rate dst_host_same_src_port_rate dst_host_srv_diff_host_rate dst_host_serror_rate
UNR-IDD	Sent_Packets Delta_Received_Packets Delta_Packets_Tx_Dropped Unknown_Load/Rate	UNR-IDD	Table_ID Packet_Match Label

A. Dataset Description

The proposed model is used to classify the threats in the network. The NSL-KDD and UNR-IDD datasets are used to carry out the tests. As stated, the NSL-KDD has a high dimensionality (41 features) that is used by the classifier, but it does not provide a better result since it causes the model to over-learn. In order to extract the optimum features, we use XGBoost. The range was reduced from 41 to 15. Selected features: 2, 3, 4, 5, 6, 8, 14, 23, 26, 29, 30, 35, 36, 37, 38. Selected features from the UNR-IDD dataset are 4, 10, 14, 19, 23, 26 and 28. A complete description of both datasets is shown in Table II.

B. Data Preprocessing

The CSA-SVM approach is employed in Data Preprocessing to enhance classification results and minimise computing time. The SVM is a widely used technology for dimension reduction. While SVM enhances in extraction of highly efficient features for representation, they are not effective for discrimination purposes. The CSA-SVM algorithm utilises an ideal projection matrix to transform a higher-dimensional feature space into a lower-dimensional feature space. This transformation retains important information necessary for data categorization. Initially, the process of data conversion is undertaken to transform the complete data collection into a standardised format. In the data filtering phase, Identify and eliminate samples or variables with high corruption risks. We usually define a threshold for missing data and eliminate subjects or variables with inaccurate data. Log processing for categorical and multi-valued discrete features involves converting and processing numeric values. This function converts symbolic and string values to numeric indexes. Index values can be donated sequentially as 0, 1, etc. Normalising data involves remapping all variables from one range to a single range. It also addresses concerns like fluctuating distribution, as certain algorithms compensate for range and distribution. However, obstacles like out-of-range values can hinder the normalisation process. Consider removing such items from the complete set if possible.

C. Feature Selection

NSL-KDD and UNR-IDD datasets contain 41 and 29 features respectively. Each feature contains a subset. To obtain low false alarms and high detection rate with high accuracy we are using these subsets. It is important to eliminate a large number of features and to examine the efficacy of the proposed approach. So, the selection of features is done. It may select the samples to form a training and testing set. The class distribution for the training set is as follows: normal - 10342, DoS - 6223, Probe - 1479, R2L - 749. The distribution of test results is as follows: the normal category has a count of 9711, DoS has a count of 7458, Probe has a count of 4421, and R2L has a count of 1094. This paper is based on the XGBoost algorithm for the selection of features.

D. Classifier Training and Testing

By using, the XGBoost algorithm the feature selection process is carried out and the set of features is trained by the SVM classifier. This classifier will identify the normal and attack classes. A testing set is then used to analyze the trained model.

E. Multiclass SVM Classifier

To detect various threats in the network, Multi-SVM classifiers are used. In this classification, the widely used techniques are "one-against-one", "one-against-all", and "Binary tree".

- The SVM classification algorithm of One-against-one requires $k(k-1)/2$ two-class SVM classifiers using all the binary pair-wise combinations of the N classes. Each classifier is trained on data from two classes.
- SVM classification Algorithm of One-against-all requires k two-class SVM classifier. Here the samples are trained with every classifier.
- Binary Tree classification it requires only $k-1$ two-class SVM classifiers for a case of k classes.

Undoubtedly, using fewer two-class classifiers improves the training and identification process. To design an intrusion detection model, a 'Binary tree' Classification is adopted.

The present research used an SVM due to the machine's inherent benefits. SVMs have great efficacy in high-dimensional domains, as exemplified in the scenario we examined. Despite the disparity between the number of dimensions and the number of samples, SVM nonetheless yields successful outcomes. SVMs exhibit memory efficiency by utilising a selected group of training points in the selection function. SVMs exhibit versatility as the selection function can be explicitly defined using various kernel features. Three SVM classifiers are created to classify the four states depending on the features of various intrusion detection styles. Actual state and the three anomaly states such as Probe, DoS and R2L. By using all samples that are trained in four states, SVM 1 is trained to differentiate between the usual and abnormal states. If the input sample of SVM 1 represents the usual state, then fix the output value of SVM 1 as +1. If the input sample of SVM 2 denotes a Dos attack, then fix the output value of SVM 2 as +1 or else -1. DoS is classified from R2L and Probing

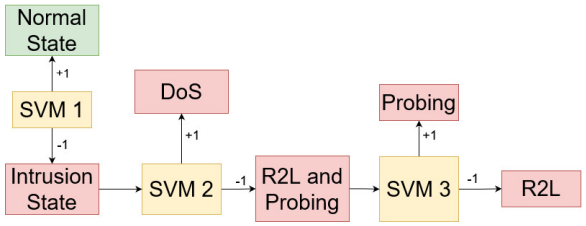

Fig. 4: Intrusion detection system using Multiclass SVM Approach

by training SVM 2. If the input sample of SVM 2 denotes the DoS attack, then fix the output value for SVM 2 as +1 or else -1. Probing and R2L are classified by training the SVM 3. If the input sample of SVM 3 denotes Probing, then fix the output value for SVM 3 as +1 or else -1. As a result, a Multi-type SVM classifier is constructed. These three Support Vector Machines implement N-RBF as a Kernel function. Their parameters C, σ and ε are improved by the Crow Search Algorithm (CSA). The most fitting parameters are those that have been adjusted to have the highest classification accuracy. Figure 4, illustrates the working process of Multi SVM classification.

F. Performance Metrics

The SVM approach optimizes performance metrics such as False Alarm Rate, F-Measure, Precision, Detection Rate, and Accuracy. Below is a detailed explanation of these performance metrics.

1) Accuracy: It is the ratio of correctly predicted observation to the total observation. The accuracy ratio of 0 specifies random guess and 1 specifies absolute accuracy.

$$A = \frac{TP + TN}{TP + TN + FP + FN}, \tag{27}$$

where TN denotes True Negative, FN denotes False Negative, TP denotes True Positive and FP denotes False Positive.

2) Precision: It is the ratio of expected positive data occurrences that are really positive and may be represented as:

$$P = \frac{TP}{TP + FP}. \tag{28}$$

3) F-Measure: The harmonic mean of recalls and precision metrics is determined as *F-Measure* and can be expressed as:

$$F = \frac{2PR}{P + R}. \tag{29}$$

4) Recall: It is the ratio of properly anticipated attack cases to the attack class's actual size, as determined by:

$$R = \frac{TP}{TP + FN}. \tag{30}$$

V. Performance Evaluation and Analysis

The key intentions of this section are as follows: (i) Efficiency of the research by evaluating Accuracy, Recall, Precision and F Score. (ii) The proposed technique is compared with other existing Algorithms. Implementation is done in MATLAB. We used the NSS-KDD and UNR-IDD datasets.

TABLE III: Performance comparison of Precision Values

Dataset	Attack Class	ANN	RNN	DNN	SVM-CSA
NSL-KDD	DoS	96.5%	95.65%	96.52%	98.01%
	Probe	75.56%	65.33%	72.15%	77.03%
	R2L	85.25%	82.00%	90.42%	97.49%
UNR-IDD	DoS	95.75%	93.37%	96.79%	97.70%
	Probe	84.75%	68.90%	79.54%	89.67%
	R2L	89.60%	81.35%	91.20%	98.42%

TABLE IV: Performance Comparison of Recall Values

Dataset	Attack Class	ANN	RNN	DNN	SVM-CSA
NSL-KDD	DoS	95.18%	87.62%	97.25%	98.56%
	Probe	88.5%	87.13%	97.90%	99.03%
	R2L	42.78%	37.97%	32.75%	43.03%
UNR-IDD	DoS	96.50 %	95.65 %	96.95 %	97.20 %
	Probe	75.56 %	69.33 %	78.25 %	87.43 %
	R2L	85.25 %	88.50 %	94.45 %	97.99 %

TABLE V: Performance comparison of F-Score values

Dataset	Attack Class	ANN	RNN	DNN	SVM-CSA
NSL-KDD	DoS	92.14%	93.48%	96.87%	97.68%
	Probe	81.14%	47.66%	71.87%	86.22%
	R2L	53.67%	33.90%	47.96%	56.83%
UNR-IDD	DoS	97.35%	96.85%	96.52%	98.01%
	Probe	78.66%	70.33%	74.45%	78.15%
	R2L	88.75%	87.45%	95.48%	97.49%

TABLE VI: Performance Comparison of Accuracy

METHODS	NSL-KDD ACCURACY	UNR-IDD ACCURACY
ANN	79.45%	90.96%
RNN	74.28%	73.30%
DNN	82.16%	77.75%
SVM-CSA	**99.58%**	**99.79%**

A. Implementation of the Proposed Method with Different Classifiers

The efficiency of various outcome parameters should be assessed to evaluate the SVM-CSA technique, and it was compared to other existing techniques such as DNN, RNN and ANN. The performance evaluations are discussed in the following order: DoS, Probe, and R2L attacks in Fig. 5, Fig. 6 and Fig. 7, respectively, based on various classification techniques. Table III, Table IV and Table V compare the performance of the proposed model's metrics to that of the other classification model. In addition, Fig. 8 examines various threats such as R2L, DoS and Probe detection accuracy. The proposed model's efficiency in detecting attacks is evaluated for each classification of attack, such as DoS, R2L, and Probe. In terms of parameters such as Accuracy, Precision, F-Score and Recall, the proposed framework detects all attacks better. For NSL-KDD and UNR-IDD datasets, the performance comparison of precision, recall, and f-score for different models with 3 attack classes are shown in Table III, Table IV and Table V, respectively. In recent times, DoS attacks have been the most difficult attacks to manage in network security domains in various fields.

The most important metrics used to assess the performance of ML algorithms are recall, accuracy, F-Score and Precision. Fig. 5 shows the analysis of a DoS attack utilising several parameters such as precision, recall, and F-score. The values obtained for the presented SVM-CSA model on the NSL-KDD dataset are 98.01%, 98.56% and 97.68%; whereas the values obtained utilizing the UNR-IDD dataset are 97.70%, 97.20% and 98.01%. Fig. 6 shows the statistical graph that compares the efficiency of the SVM-CSA classifier-based methodology using the same metrics. In the case of Precision metric from Table III, the classical SVM-CSA approach is revealed to be slightly high on detecting for various classes such as DoS (98.01% and 97.70%), Probe (77.03% and 89.67%) and R2L (97.49% and 98.42%) when compared to other models. The efficiency of the SVM-CSA classifiers in detecting various attacks is shown in Figs. 3 and 4. Fig. 7 shows the analysis of R2L attack detection, incorporating several characteristics such as precision, recall, and F-score. The corresponding values for these parameters are 97.49%, 43.03% and 56.83% utilizing the NSL-KDD dataset; and 98.42%, 97.99% and 97.49% using the UNR-IDD dataset for the SVM-CSA approach. Finally, it is evident from the graph that the application of our proposed method yields better performance than ANN, RNN and DNN.

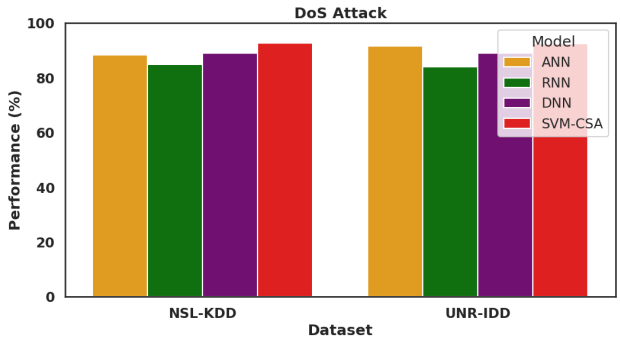

Fig. 5: DoS attack detection performance analysis

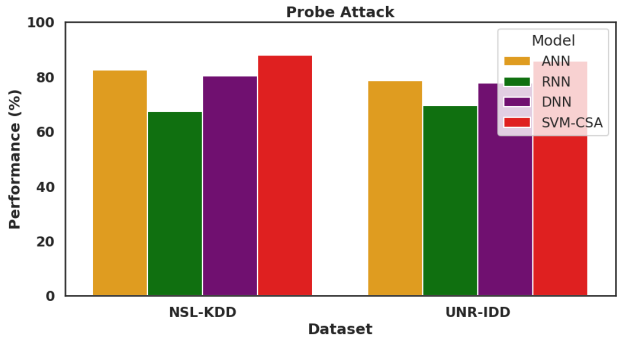

Fig. 6: PROBE attack detection performance analysis

Table IV shows how well the proposed model executed in each attack class. Existing techniques were compared to the proposed technique. When compared to RNN (87.62% and 95.65%), the proposed methods recall was far superior to ANN (95.18% and 96.50%) and DNN (97.25% and 96.95%) for DoS attack. When comparing for Probe class efficiency performed is close to each but the highest is outperformed by the proposed method (99.03%, 87.43%). When coming to R2L related to Probe (43.03% and 97.99%), it is far less than the detection of the other two attacks.

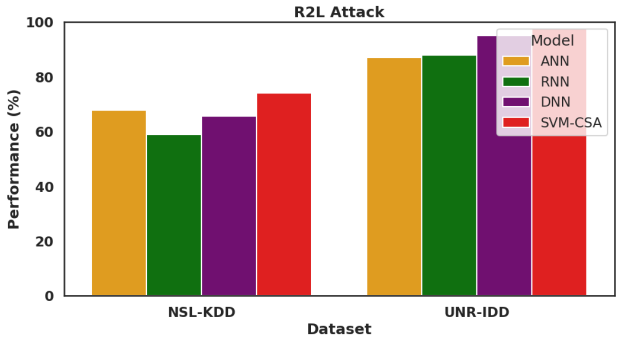

Fig. 7: R2L attack detection performance analysis

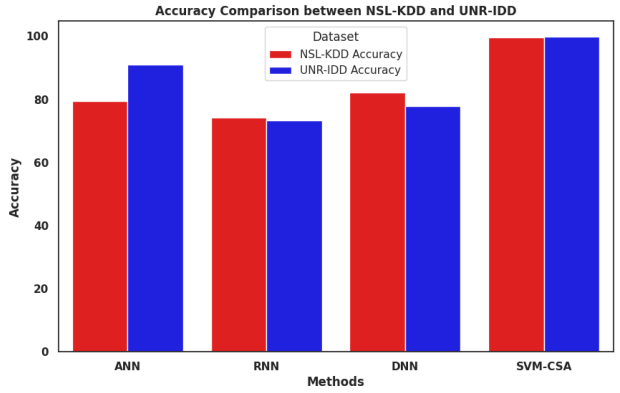

Fig. 8: Accuracy result of different classifiers

In the case of F-Score, as shown in Table V, it is observed that all the attacks using CSA-SVM show higher values than the ANN (Dos: 92.14%, Probe: 81.14% and R2L: 53.67%), RNN (DoS: 93.48%, Probe: 47.66% and R2L: 33.90%) and DNN (DoS: 96.87%, Probe: 71.87% and R2L: 47.96%) for the NSL-KDD dataset. Using the UNR-IDD dataset the proposed method shows the higher values from ANN (DoS: 97.35%, Probe: 78.66% and R2L: 88.75%), RNN (DoS-96.85%, Probe: 70.33% and R2L: 87.45%) and DNN (DoS: 96.52%, Probe: 74.45% and R2L: 95.48%).

According to the information presented in Table VI, the proposed method has an accuracy level of 99.58% for NSL-KDD dataset and 99.79% accuracy level for UNR-IDD dataset, while the other three procedures being examined had accuracy levels of 79.45%, 74.28%, 82.16% for NSL-KDD dataset and 90.96%, 73.30%, 77.75% for UNR-IDD dataset. Based on the accuracy graph, it is evident that the proposed technique, as presented in Fig. 8, shows an ongoing pattern of improvement in accuracy. As compared to other approaches, the experiments conducted using the Real-Time Deep Extreme Learning Machine (RTS-DELM) technique [40] is proving to be highly effective and obtaining 93.58% and 6.42% accuracy and miss rate collectively. The combination of SVM and Aquila Optimizer is employed for intrusion detection. Comparative investigation demonstrates that the detection accuracy in the wireless network is 98.76% [41]. Another significant investigation in the field of intrusion detection utilising Genetic Algorithms (GA) is recorded in [42]. This study developed a Genetic Programming-Support Vector Machine (GPSVM) by effectively combining genetic programming and SVM. The observed detection rates exhibited variability based on the characteristics of the attacks. The GPSVM system achieved a detection rate of 89.28%, demonstrating exceptional proficiency in identifying U2R attacks. By contrast, a recent study described in [43] focused on the use of a real-time sequential deep extreme learning system for intrusion detection. The machine demonstrated its exceptional ability by obtaining a peak accuracy rate of 93.58% during testing on a combined dataset consisting of both NSL-KDD and KDD CUP 99. This study highlights the importance of taking into account various approaches and datasets while seeking efficient intrusion detection algorithms. Among the latest studies on intrusion detection with an accuracy of 93.58%, [44] stands out as a valuable addition to the existing study landscape. The detailed methods and findings of this work necessitate a thorough analysis to extract valuable insights and prospective breakthroughs in the field. The extension of previous work [45], an optimization technique using the Gravitational Search Algorithm (GSA) to improve the Fuzzy Inference System (FIS) for attack detection. pre-processes packets, clusters them through Possibilistic Fuzzy C-Means (PFCM), and feeds them into an adaptable fuzzy inference framework. The GSA enhances the fuzzy framework to detect DoS, R2L, U2R, and Probe anomalous data. Applies Probability Mass Function (PMF) and Min-Max data pre-processing approaches in Deep Belief Networks (DBNs) to minimize the training process function and prevent feature homogeneity and network overloading. According to the previous related work, we can say that our proposed work detects DoS, Probe, and R2L attacks better than ANN, RNN, and DNN. The proposed method outperforms other classifiers with 99.58% and 99.70% accuracy for both NSL-KDD and UNR-IDD datasets. SVM-CSA outperforms other approaches in precision, recall, and F-score for DoS, Probe, and R2L attack classes. Thus, the SVM-CSA model outperforms the previous research and methods in accuracy, precision, recall, and F-score.

VI. Conclusions and Future Work

In cloud computing, system security has become one of the primary concerns due to the variety of cyberattacks on networks. As a matter of fact, intrusion detection has become a vital part of system security. In this paper, we proposed XGBoost and SVM-CSA optimization techniques to detect the threats in the cloud. XGBoost Algorithm is used for the selection of features from the dataset followed by Support Vector Machine for intrusion classification by using the NSL-KDD and UNR-IDD datasets. Besides, SVM employs the Crow Search Algorithm to optimize the Kernel Parameter in order to maximize classification performance. The results are evaluated by comparing our proposed technique with the other existing algorithms such as RNN, DNN and ANN. This analysis concludes that SVM achieves the highest percentage in Accuracy, Precision, Recall and F-Score. In the future study, we intend to develop different classification algorithms combined with optimization methods for reducing false alarm rates while detecting attacks.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Grant No. 62071327) and the Tianjin Science and Technology Planning Project (Grant No. 22ZYYYJC00020).

REFERENCES

[1] D. Javeed *et al.*, "An explainable and resilient intrusion detection system for industry 5.0," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 1342–1350, 2024.

[2] W. Z. Khan *et al.*, "Communal acts of iot consumers: A potential threat to security and privacy," *IEEE Transactions on Consumer Electronics*, vol. 65, no. 1, pp. 64–72, 2018.

[3] A. U. Rehman *et al.*, "Enhancement in quality-of-services using 5g cellular network using resource reservation protocol," *Physical Communication*, vol. 55, p. 101907, 2022.

[4] M. Yamauchi and othersi, "Anomaly detection in smart home operation from user behaviors and home conditions," *IEEE Transactions on Consumer Electronics*, vol. 66, no. 2, pp. 183–192, 2020.

[5] M. Bakro *et al.*, "Efficient intrusion detection system in the cloud using fusion feature selection approaches and an ensemble classifier," *Electronics*, vol. 12, no. 11, p. 2427, 2023.

[6] J. Ali *et al.*, "An intelligent blockchain-based secure link failure recovery framework for software-defined internet-of-things," *Journal of Grid Computing*, vol. 21, no. 4, p. 57, 2023.

[7] J. Ali, B.-h. Roh, B. Lee, J. Oh, and M. Adil, "A machine learning framework for prevention of software-defined networking controller from ddos attacks and dimensionality reduction of big data," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2020, pp. 515–519.

[8] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert systems with applications*, vol. 148, p. 113249, 2020.

[9] D. Javeed *et al.*, "An intelligent intrusion detection system for smart consumer electronics network," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 4, pp. 906–913, 2023.

[10] C. K. Dehury, S. N. Srirama*, P. K. Donta, and S. Dustdar, "Securing clustered edge intelligence with blockchain," *IEEE Consumer Electronics Magazine*, vol. 13, no. 1, pp. 22–29, 2024.

[11] B.-C. Choi *et al.*, "Secure firmware validation and update for consumer devices in home networking," *IEEE Transactions on Consumer Electronics*, vol. 62, no. 1, pp. 39–44, 2016.

[12] I. Murturi *et al.*, "Learning-driven zero trust in distributed computing continuum systems," in *2023 Intl Conf on Cyber Science and Technology Congress*. IEEE, 2023, pp. 0044–0049.

[13] C. Bicer, I. Murturi, P. K. Donta, and S. Dustdar, "Blockchain-based zero trust on the edge," *arXiv preprint arXiv:2311.16744*, 2023.

[14] M. Bakro *et al.*, "An improved design for a cloud intrusion detection system using hybrid features selection approach with ml classifier," *IEEE Access*, vol. 11, pp. 64 228–64 247, 2023.

[15] S. S. Gill *et al.*, "Modern computing: Vision and challenges," *Telematics and Informatics Reports*, vol. 13, p. 100116, 2024.

[16] Y. Liu *et al.*, "Ieee p2668-compliant multi-layer iot-ddos defense system using deep reinforcement learning," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 1, pp. 49–64, 2022.

[17] S. I. Shyla and S. Sujatha, "Cloud security: Lkm and optimal fuzzy system for intrusion detection in cloud environment," *Journal of Intelligent Systems*, vol. 29, no. 1, pp. 1626–1642, 2019.

[18] D. Wang and G. Xu, "Research on the detection of network intrusion prevention with svm based optimization algorithm," *Informatica*, vol. 44, no. 2, 2020.

[19] A. Sharma *et al.*, "Homlc-hyperparameter optimization for multi-label classification of intrusion detection data for internet of things network," *Sensors*, vol. 23, no. 19, p. 8333, 2023.

[20] R. SaiSindhuTheja and G. K. Shyam, "An efficient metaheuristic algorithm based feature selection and recurrent neural network for dos attack detection in cloud computing environment," *Applied Soft Computing*, vol. 100, p. 106997, 2021.

[21] R. Rajendran, S. Santhosh Kumar, Y. Palanichamy, and K. Arputharaj, "Detection of dos attacks in cloud networks using intelligent rule based classification system," *Cluster Computing*, vol. 22, pp. 423–434, 2019.

[22] Ü. Çavuşoğlu, "A new hybrid approach for intrusion detection using machine learning methods," *Applied Intelligence*, vol. 49, pp. 2735–2761, 2019.

[23] V. Balamurugan and R. Saravanan, "Enhanced intrusion detection and prevention system on cloud environment using hybrid classification and ots generation," *Cluster Computing*, vol. 22, no. Suppl 6, pp. 13 027–13 039, 2019.

[24] A. N. Jaber and S. U. Rehman, "Fcm–svm based intrusion detection system for cloud computing environment," *Cluster Computing*, vol. 23, pp. 3221–3231, 2020.

[25] M. S. Taj *et al.*, "Enhancing anomaly based intrusion detection techniques for virtualization in cloud computing using machine learning," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 18, no. 5, pp. 68–78, 2020.

[26] P. Devan and N. Khare, "An efficient xgboost–dnn-based classification model for network intrusion detection system," *Neural Computing and Applications*, vol. 32, pp. 12 499–12 514, 2020.

[27] B. Hajimirzaei and N. J. Navimipour, "Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm," *Ict Express*, vol. 5, no. 1, pp. 56–59, 2019.

[28] S. R. K. Tummalapalli and A. Chakravarthy, "Intrusion detection system for cloud forensics using bayesian fuzzy clustering and optimization based svnn," *Evolutionary Intelligence*, vol. 14, pp. 699–709, 2021.

[29] G. S. Kushwah and V. Ranga, "Voting extreme learning machine based distributed denial of service attack detection in cloud computing," *Journal of Information Security and Applications*, vol. 53, p. 102532, 2020.

[30] L. Lv *et al.*, "A novel intrusion detection system based on an optimal hybrid kernel extreme learning machine," *Knowledge-based systems*, vol. 195, p. 105648, 2020.

[31] S. I. Shyla and S. Sujatha, "An efficient automatic intrusion detection in cloud using optimized fuzzy inference system," *International Journal of Information Security and Privacy*, vol. 14, no. 4, pp. 22–41, 2020.

[32] Q. Tian *et al.*, "An intrusion detection approach based on improved deep belief network," *Applied Intelligence*, vol. 50, pp. 3162–3178, 2020.

[33] A. Tharwat and A. E. Hassanien, "Quantum-behaved particle swarm optimization for parameter optimization of support vector machine," *Journal of Classification*, vol. 36, pp. 576–598, 2019.

[34] R. M. Yadav, "Effective analysis of malware detection in cloud computing," *Computers & Security*, vol. 83, pp. 14–21, 2019.

[35] V. Kanimozhi and T. P. Jacob, "Artificial intelligence outflanks all other machine learning classifiers in network intrusion detection system on the realistic cyber dataset cse-cic-ids2018 using cloud computing," *ICT Express*, vol. 7, no. 3, pp. 366–370, 2021.

[36] P. Mishra *et al.*, "Vmshield: Memory introspection-based malware detection to secure cloud-based services against stealthy attacks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 6754–6764, 2021.

[37] T. Thilagam and R. Aruna, "Intrusion detection for network based cloud computing by custom rc-nn and optimization," *ICT Express*, vol. 7, no. 4, pp. 512–520, 2021.

[38] S. Dwivedi *et al.*, "Building an efficient intrusion detection system using grasshopper optimization algorithm for anomaly detection," *Cluster Computing*, pp. 1–20, 2021.

[39] C. Liu, Z. Gu, and J. Wang, "A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning," *Ieee Access*, vol. 9, pp. 75 729–75 740, 2021.

[40] M. P. N. Kalavadekar and S. S. Sane, "Building an effective intrusion detection system using genetic algorithm based feature selection," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 16, no. 7, 2018.

[41] L. Abualigah *et al.*, "Modified aquila optimizer feature selection approach and support vector machine classifier for intrusion detection system," *Multimedia Tools and Applications*, pp. 1–27, 2024.

[42] M. S. M. Pozi, M. N. Sulaiman, N. Mustapha, and T. Perumal, "Improving anomalous rare attack detection rate for intrusion detection system using support vector machine and genetic programming," *Neural Processing Letters*, vol. 44, pp. 279–290, 2016.

[43] T. Ghazal, "Data fusion-based machine learning architecture for intrusion detection," *Computers, Materials & Continua*, vol. 70, no. 2, pp. 3399–3413, 2022.

[44] M. A. Khan *et al.*, "A machine learning approach for blockchain-based smart home networks security," *IEEE Network*, vol. 35, no. 3, pp. 223–229, 2020.

[45] J. K. Samriya *et al.*, "Adversarial ml-based secured cloud architecture for consumer internet of things of smart healthcare," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 2058–2065, 2023.