Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.053721

ARTICLE

GRU Enabled Intrusion Detection System for IoT Environment with Swarm Optimization and Gaussian Random Forest Classification

Mohammad Shoab* **and Loiy Alsbatin***

Department of Computer Science, College of Science and Humanities Al Dawadmi, Shaqra University, Al Dawadmi, 17441, Saudi Arabia

*Corresponding Authors: Mohammad Shoab. Email: mshoab@su.edu.sa; Loiy Alsbatin. Email: lalsbatin@su.edu.sa

Received: 08 May 2024 Accepted: 20 August 2024 Published: 15 October 2024

ABSTRACT

In recent years, machine learning (ML) and deep learning (DL) have significantly advanced intrusion detection systems, effectively addressing potential malicious attacks across networks. This paper introduces a robust method for detecting and categorizing attacks within the Internet of Things (IoT) environment, leveraging the NSL-KDD dataset. To achieve high accuracy, the authors used the feature extraction technique in combination with an auto-encoder, integrated with a gated recurrent unit (GRU). Therefore, the accurate features are selected by using the cuckoo search algorithm integrated particle swarm optimization (PSO), and PSO has been employed for training the features. The final classification of features has been carried out by using the proposed RF-GNB random forest with the Gaussian Naïve Bayes classifier. The proposed model has been evaluated and its performance is verified with some of the standard metrics such as precision, accuracy rate, recall F1-score, etc., and has been compared with different existing models. The generated results that detected approximately 99.87% of intrusions within the IoT environments, demonstrated the high performance of the proposed method. These results affirmed the efficacy of the proposed method in increasing the accuracy of intrusion detection within IoT network systems.

KEYWORDS

Machine learning; intrusion detection; IoT; gated recurrent unit; particle swarm optimization; random forest; Gaussian Naïve Bayes

1 Introduction

The Global Internet Statistics Report has confirmed a recent surge in Internet usage, revealing 4.66 billion active users and more than two quintillion bytes of data generated daily. This exponential increase reflects a dramatic rise in the availability of data from a variety of sources, underpinned by an impressive progress in data collection and handling methodologies and hacking tools. This generates a critical need for enhanced privacy and data security measures to thwart a variety of malicious attacks and intrusions of different forms. In practice, the sheer volume and the increasing speed with which the data arrive make it difficult for traditional intrusion detection systems to effectively and efficiently identify intrusions or attacks, and to which pre-data science intrusion detection models are largely inapplicable [1]. However, all these data are often unmanageable by traditional methods or techniques,

which means that advanced intelligent methodologies and sophisticated computational approaches are required. Intrusion Detection Systems (IDSs) play a key role in the context in which they are developed for the Internet of Things (IoT). IoT connects various devices, from home appliances to industrial machines, enabling them to share data and work more efficiently. However, this vast network of devices also presents significant security challenges. An intrusion detection system (IDS) is crucial for safeguarding IoT networks because these devices are often vulnerable to cyberattacks due to their diverse and resource-constrained nature. IDS helps by continuously monitoring network traffic for suspicious activities, allowing for real-time threat detection and response. It ensures data privacy, compliance with regulations, and protection against unauthorized access, making our increasingly connected world safer and more reliable. If such activities are identified, alarms are often sent to network system administrators so that they can properly manage and classify those attacks or intrusions. There are numerous ML algorithms that can be utilized to perform such kinds of tasks [2,3].

In conclusion, based on the different reviews, the process of an IDS is divided into four phases: 1) data pre-processing, 2) feature extraction, 3) feature selection, and 4) classification. Initially, data pre-processing is performed to make the data standardized as much as possible. It eases the way the data is prepared for specialized planning, training, and testing in the next steps [4]. Then, the pre-processed data is fed into the feature extraction step, where raw data are transformed into numerical features. These features can summarize and represent the information from the original dataset, which, given several resemblances between the inputs, will in return reduce redundancy within the original dataset [5]. Therefore, in our proposed model, feature extraction is implemented by integrating the gated recurrent unit (GRU) with an autoencoder. The GRU method addresses vanishing gradient issues inherent in recurrent neural networks, thereby enhancing memory capacity [6]. Similarly, the autoencoder is utilized for data encoding and decoding, contributing to the classification of normal and abnormal events [7]. In addition, to effectively deal with problems related to the elimination of redundant and irrelevant data, a feature selection process is also adopted. The feature selection process is applied to enhance learning accuracy, reduce computational time, and facilitate a better understanding of the data [8]. Therefore, a feature selection is proposed by the integration of cuckoo search and PSO as a cuckoo-search algorithm, which is inspired by the brood parasitic habits of some cuckoo species with their leveled eggs, brood partitioning, and Levy flights random walk is integrated with the PSO, in which a population of particles form a search technique to optimal results are obtained [9]. Since each of these two algorithms has exquisite characteristics, the other has defects, the blend of scrounger possessing and PSO to choose features in a more energetic way [10,11]. The proposed framework ultimately produces classified data by some designated technique, which is a significant part of an intrusion detection system (IDS) [12]. In supervised learning, classification assigns new observations to the category or class to which they most likely belong, based on training data. An appropriate classification algorithm is necessary to improve the accuracy of an IDS. Consequently, random forest combined with Gaussian Naïve Bayes is employed in our model as a classifier. In the proposed model, the random forest, a collection of decision trees trained with the bagging method [13,14], is combined with the Gaussian Naïve Bayes classifier, to provide the best class label through selected features. This better performance shows that the proposed IDS can be utilized in real-world applications. The main contributions of this paper can be summarized as follows:

- Achievement of reliable feature extraction using the proposed Auto-encoder with Gated Recurrent unit technique.
- Selection of elegant features with the proposed Cuckoo search (CS) integrated with the Particle Swarm Optimization model.

- Classification of processed features with the newly developed random forest with Gaussian Naïve Bayes (RF-GNB).
- Estimate the effectiveness of the proposed system to identify attacks in the IoT environment and compare it with the state-of-the-art methods.

The remainder of this paper is organized as follows: Section 2 emphasizes the related works about the intrusion detection system and the different implemented methods introduced for addressing the various kinds of attacks. Further, the proposed GRU-Auto Encoder using the RF-GNB model has been elaborated briefly in Section 3. Moreover, in Section 4, the results and the discussion have been exhibited and demonstrated. Lastly, in Section 5, the paper has concluded with the future work.

2 Related Works

In recent years, the proliferation of threat attacks in the IoT environment has become increasingly apparent. Traditional network intrusion detection systems, relying on feature filtering, exhibit certain limitations that hinder their efficacy in identifying emerging threats within the given timeframe. In response, the authors of [15] proposed an adaptive ensemble learning model, leveraging the NSL-KDD dataset, to address both the contemporary challenges and past shortcomings encountered in intrusion detection technology. The model consists of a multi-tree algorithm integrating a carefully analyzed training data distribution to develop an algorithm. Detection efficacy is further improved through the choice of diverse base classifiers and the development of an ensemble adaptive mechanism based on a voting algorithm. This approach was validated with the NSL-KDD dataset, which led to an accuracy rate of 84.2% for the MultiTree algorithm and 85.2% for the full-designed algorithm. It is recommended that they work on future optimization of feature selection and data preprocessing to detect intrusion better. In addition, they are also exploring deep learning techniques to enhance network longevity. For example, in [16], a deep neural network (DNN) is used to design a powerful and flexible IDS to detect and classify normal and unknown irregular cyber-attacks. The developed DNN was able to outperform old-fashioned techniques, monitoring the network traffic with remarkable efficacy and signaling the hosts to potential cyber-threats ahead of the attack. The authors of [2] presented a technique that is a hybrid of deep learning and traditional techniques to design a Network IDS (NIDS). The methods were tested with the KDD CUP'99 and NSL-KDD dataset and were concluded with efficient solutions of minimizing the complexity of the frames, and maximizing the attack detection accuracy. It is necessary to design these systems with new strategies, light implementations, and develop NIDS based deep learning systems to ensure effective intrusion detection systems within the network in the time ahead [2,16].

The research presented in [5] introduced a model employing a stacked sparse autoencoder (SSAE), representing an example of deep learning techniques, that extract high-level feature representations of information related to intrusive behavior. This allows for the automatic extraction of deep-sparse features that are crucial for detecting initial occurrences. The efficacy of SSAE underscores the feasibility and efficiency of using deep learning to perform feature extraction; in fact, it is demonstrated in both multiclass and binary classification tasks. This model is particularly important as it enhances the efficacy of intrusion detection systems; as described in [5], it not only provides a higher classification accuracy, but also reduces the training time in the machine learning process. An autoencoder intrusion detection system (AE-IDS) amped with a random forest algorithm has been proposed in [13]. This system trains the intrusion dataset in order to acquire features and then groups them. The experimental results demonstrate a decrease in processing time and an increase in prediction accuracy, which outperforms simple learning methods [17,18].

Utilizing machine learning as a research methodology proves apt for discerning network threats. In [19], two machine learning algorithms are developed with feature extraction to classify anomalous activities within network traffic. The IDS is investigated through an aggregation of PSO and decision tree algorithm, as well as PSO with k-nearest neighbor algorithm, aiming to innovate design strategies. This recommended approach undergoes evaluation using metrics such as recall, specificity, accuracy, and consistency in cybersecurity databases. The results underscore that the PSO and k-nearest neighbor (KNN) algorithm outperforms the PSO and decision tree (DT) algorithm in identifying network attacks. This study's applicability extends to other network intrusion datasets, suggesting future enhancements through the incorporation of deep learning algorithms into IDS. Similarly, in [20], the PSO algorithm is implemented with feature selection to enhance IDS accuracy and detection rates. The results indicate superior performance and accuracy rates compared to conventional algorithms. Future discussions should explore employing a larger dataset for feature selection techniques to further enhance the accuracy rates of the proposed approach [21,22].

The majority of classifiers exhibited excessively high accuracies. In [23], an examination of the performance of classifier mixtures was carried out which consisted of different types such as Support Vector Machine (SVM), k-Nearest Neighbors (KNN), Naïve Bayes, and Decision Tree, and this collaborative approach was applied to different attack types such as Normal attacks and Denial of Service (DoS) attacks, and the performance of different classifiers were evaluated by KDDCup99 dataset based on the accuracy and confusion matrices, and the results showed that the classifier models were not so efficient for Normal but they showed good improvement for DoS attacks, and it can be applied on other datasets would make them more effective in detection. Intrusion detection was similarly the subject of research. In [11], a novel classification technique was introduced to identify network or host abnormalities, employing combined datasets from UNSW-NB15 and KDD-Cup'99 implemented in Particle Swarm Optimization (PSO) and Random Forest algorithms. Performance evaluation, including a confusion matrix comparison, revealed a detection rate of 75.94% and an accuracy rate of 97%. This methodology demonstrated significant learning enhancement and satisfactory accuracy. Likewise, authors in [24] proposed a combined algorithm comprising SVM, Decision Tree, and Naïve Bayes to predict unwanted information, aiming to mitigate false alarm rates. Utilizing the WEKA tool for statistical reporting, the study concluded that Decision Tree outperformed the other classifiers. While numerous anomaly detection techniques have been introduced, their effectiveness in meeting real-time requirements remains a challenge. Hence, in [25], a new detection algorithm named LGMAD was introduced for real-time attack detection, integrating Long-short Term Memory (LSTM) and Gaussian Mixture Model (GMM). The evaluation utilized datasets such as NAB and self-made data, with future research opportunities lying in comparing LGMAD against existing detection algorithms to demonstrate its superiority.

The different challenges or limitations have been analyzed while undergoing the traditional work listed below:

- Numerous ML techniques have been used to design the IDS for guarding network integrity and privacy [26]. Finally, as it was emphasized in [19], recent research showed the ability of deep learning to detect malicious activities.
- There exists a pressing need to enhance accuracy rates in IDS. For instance, in [20], from the dataset of NSL-KDD, some 10 features by their appropriate feature selection method were picked which proves that a higher number of features will enhance the accuracy rates of the system.

- IDSs deployed in diverse environments demonstrate distinct accuracy rates. For instance, an accuracy rate of 97% was reported in [11]. Further improvement in accuracy rates is essential for better detection of occurring attacks.
- Similarly, the methodology proposed by the author in [21] introduces modifications to the cuckoo search algorithm, yielding promising results with an accuracy rate of 98.16% and a detection time of approximately 96.83%. Nevertheless, there remains room for improvement in performance metrics to ensure precise intrusion detection system functionality, a direction warranting further exploration in this study.

3 Methodology

A newly proposed technique is presented in this section. The proposed technique gives a detailed overview in Fig. 1. First, NSL-KDD dataset is loaded and pre-processing is performed. Feature selection and extraction are performed to optimize the accuracy. Feature extraction is performed with Auto Encoder GRU, which only extracts significant features. The extracted set is then processed using a variational cuckoo swarm algorithm for important features extraction. The anomaly is classified with the hybridization of Gaussian Naïve Bayes and the random forest algorithm is applied to the selected features. The features are then fed into the prediction phase to distinguish between normal and abnormal data. Finally, the proposed model is evaluated with a number of performance metrics and compared with existing models to demonstrate its efficiency and effectiveness.

3.1 Data Pre-Processing

This is the first phase of the suggested methodology, with an emphasis on pre-processing the data. Transforming raw data into a format conducive to efficient utilization in subsequent processing stages has been done in this phase. Initially, the data undergoes a cleaning process utilizing the min-max method. This normalization aids in standardizing the scale of the training data, typically ranging between 0 and 1, thereby enhancing training data coherence. Subsequent to the data cleaning procedure, the refined data obtained from the preceding step is fed into the feature extraction process.

3.2 Feature Extraction Using the GRU-Auto Encoder Model

The hybridization of an autoencoder and a GRU leverages the strengths of both techniques for effective feature extraction in machine learning models. The autoencoder, comprising an encoder and decoder, reduces data dimensionality by learning compressed representations, thus capturing essential features. Meanwhile, the GRU, a type of recurrent neural network, excels in processing sequential data by maintaining context through its gating mechanism. Integrating these two approaches allows for the extraction of significant, time-dependent features from complex datasets, enhancing the model's ability to differentiate between normal and abnormal events with improved accuracy and efficiency. Utilizing the gated recurrent unit in conjunction with the autoencoder model allows for the efficient reconstruction of data abnormalities across large sequences by incorporating data from previous moments. This learning approach is driven by the objective of discerning shifts from normal to abnormal states, thereby enhancing the accuracy of condition identification during abnormal operations. The architectural configuration of the hybrid GRU-autoencoder model is illustrated in Fig. 2.

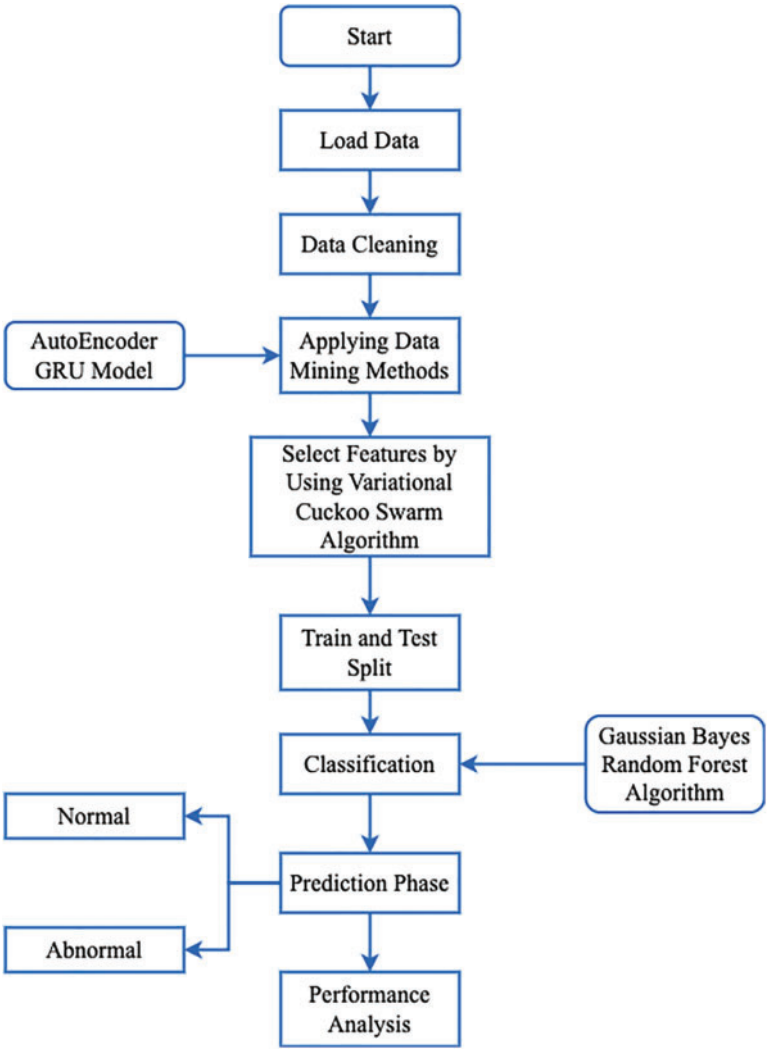

Figure 1: Flowchart of the proposed model

During the recent time period denoted as t, data connected to each parameter of the steam drum is evaluated by progressing $t-1, t-2, \ldots, t-p$ in a forward manner, while data provided at $t+1, t+2, \ldots, t+q$ is evaluated backward. This alteration transforms the provided data into multivariable form. The preceding $t-p$ data step-time serves as the input, and the outcome corresponds to $t+q$ data based on the step-time iteratively rolled. This adjustment in the data learning process for the steam drum facilitates the dynamic characterization of abnormal operating conditions, with p and q being dependent on the sampling time. The parameters for the provided data used in the model analysis under abnormal operating conditions are denoted as $PIZA$, $FIYC$, and $T1$, and these parameters are represented as equations detailed below:

$$T1\,(t-p), PIZA\,(t-p), \ldots, FIYC\,(t-P), \ldots, T1\,(t-p+1), PIZA(t-p+1), \ldots, FIYC\,(t-p+1), \ldots, T1\,(t), PIZA\,(t), \ldots, PIZA\,(t), \ldots, T1\,(t+1), PIZA(t+1), \ldots, FIYC\,(t+1), \ldots, T1\,(t+q), PIZA \quad (1)$$

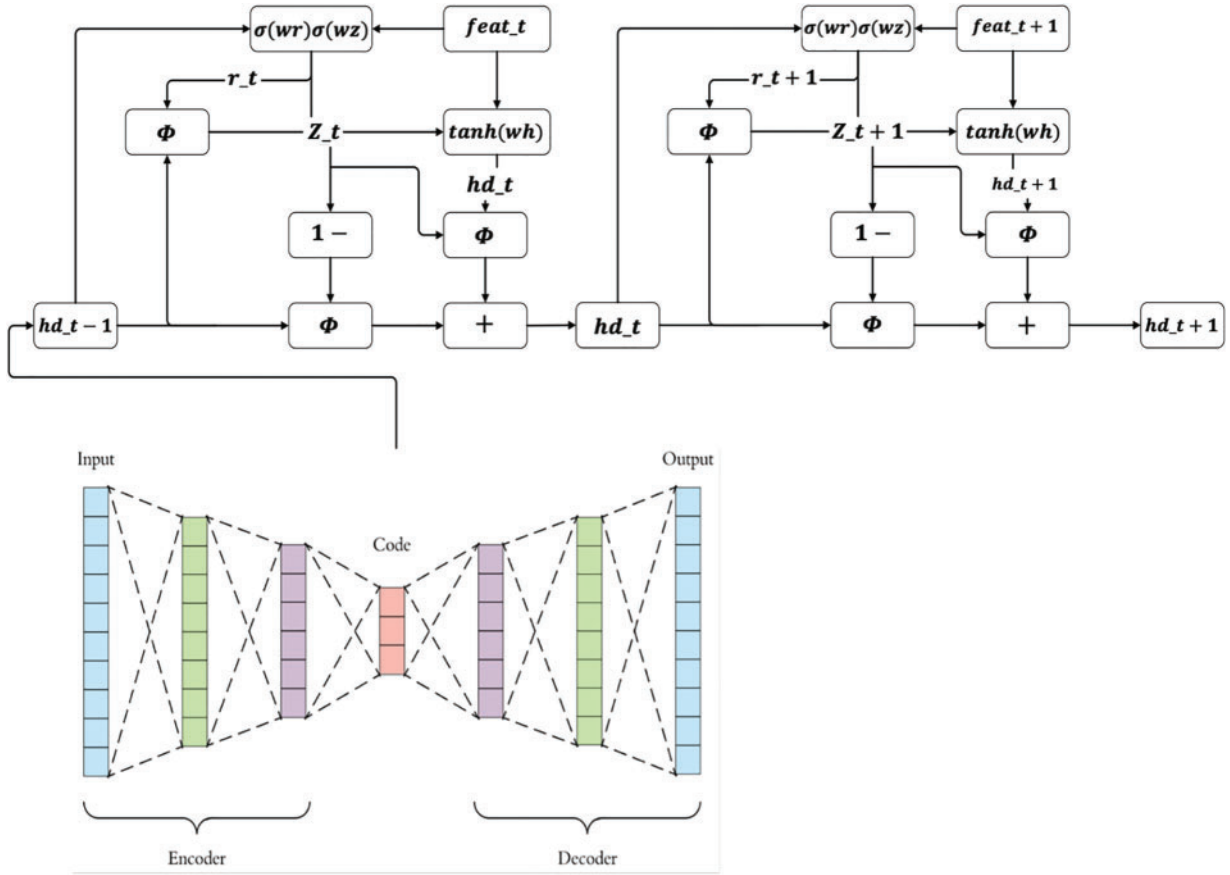

Figure 2: Hybridization of Auto Encoder and GRU

The process of reconstruction of the irregular models of the steam drum while utilizing the GRU-Auto Encoder network is emphasized as follows.

Stage 1: the mapping of non-linear parameters has performed in the contributed data $feat_t$ to get the encoder output as $u_{t,1}$:

$$u_{t,1} = ReLU(feat_t, h_{t-1}) \tag{2}$$

Here, the $ReLU$ is the function activation of non-linear, h_{t-1} is the information state of the preceding time.

Stage 2: The non-linear mapping transmits Encoder output while the process of decoding:

$$u_{t,m} = ReLU(linear(u_{t,m-1}, h_{t-1})) \tag{3}$$

$$m = 2 \ldots, M \tag{4}$$

where M represented the total number of hidden layers, $u_{t,m}$ is considered as the decoder output data.

Stage 3: linear mapping performed in the output $u_{t,M}$ of the encoder to obtain the reconstruction of data $feat'_t$:

$$feat'_t = linear(u_{t,m}, h_{t-1}) \tag{5}$$

Stage 4: The method of layers of hidden is updated is characterized as given below:

$$h_t = GRU(feat_t, h_{t-1}, u_{t,m}) \tag{6}$$

Mean Square Error (MSE) considered as the loss function for the GRU-Auto Encoder model:

$$Loss_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(feat_i - feat_i') \tag{7}$$

The error of the loss function is minimized by optimizing the variables in the structure of the network.

3.3 Feature Selection Using Variational Cuckoo-Particle Swarm Optimization Algorithm

The Cuckoo Search Methodology (CSM) is an evolutionary algorithm characterized by a search strategy technique based on the parasitism behavior of certain cuckoo species, which lay their eggs in the nests of other bird species, known as hosts. If the host bird detects these eggs as foreign, it can abandon its nest or discard the alien eggs and relocate them to a new nest. In the CS technique, each species of cuckoo adjusts its position over time, and each egg in a nest contributes to only one new outcome of the solution.

PSO is a stochastic algorithm utilized for optimizing searches by leveraging the collective intelligence of a swarm. This approach draws inspiration from the examination of various collective behaviors observed in living species, such as fish schooling, bird flocking, and swarm theory. PSO operates by employing a search strategy based on a population of individuals referred to as particles. Typically, in PSO, these particles navigate through multidimensional search spaces. Each particle is assessed based on its particular velocity and the functional objectives that influence its movement. Fig. 3 depicts the visual representation of PSO algorithm, and some main characteristics of PSO are listed below:

- Swarm Intelligence
- Population-Based Search
- Velocity and Position Update
- Global and Local Search

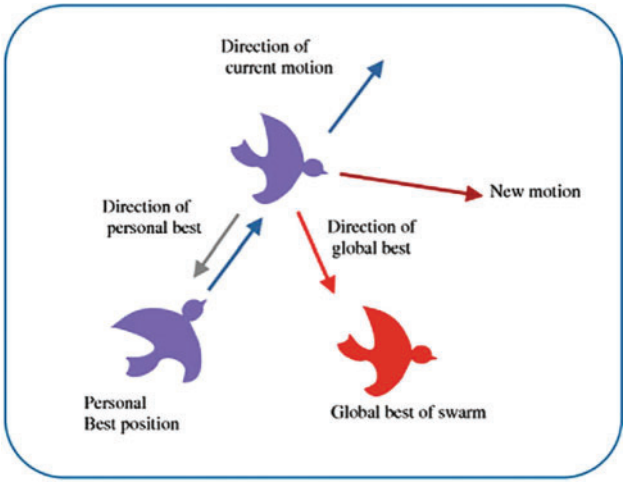

Figure 3: Particle swarm optimization

- Continuous Adaptation
- Fitness Evaluation
- Convergence Behavior
- Parameter Sensitivity
- Parallelization Potential

The position vector and velocity updated and obtained as the below equation:

$$v_i^{p+1} = W^p * v_i^p + f_1 * r_1 * \left(pbest_i^p - a_i^p\right) + f_2 * r_2 * \left(gbest_i^p - a_i^p\right) \tag{8}$$

$$a_i^{p+1} = a_i^p + v_i^{p+1} \tag{9}$$

$$w^t = \frac{M_{gen} - p}{M_{gen}} \tag{10}$$

Here a_i^p and v_i^p denotes the velocity and the position of the particle at the given period t, W_p is the inertial weight, M_{gen} is the maximum generation of the technique, f_1 as well as f_2 are the non-changeable acceleration in the positive manner and the r_1 and r_2 are the random values that are generated in the range of [0,1]. $pbest_i$ is the relevant outcome of the i^{th} output as well as the $gbest$ is the tracked best solution by any of the element in each generation of swarm. To optimize the more feature selection, the fresh attribute has involved, namely the variation attribute, for attaining the optimum solution. These changes of the search able to attain the search in an optimal way. Algorithm 1 denotes the procedure of the variational CS-POS technique for the improvement of the optimal way that denoted below. The time and space complexity for the below implemented algorithm is given as $O(N*T*D)$ and $O(N*D)$, respectively. In this, N represented as the population size, D as dimension for search and T is the Iteration time. Moreover, the below stated algorithm is used for selecting the algorithm and the computation time for this algorithm is around 5.8 ms.

Algorithm 1: Variational CS-PSO algorithm

Begin

Objective function $E(a), a = (a_1, I, a_d)^P$

Obtain the primary on n host nests population $x_i\ (i = 1, 2, \ldots, n)$ and corresponding random velocities

Calculate fitness value E_i

Obtain p_{best_i} *and* g_{best}

While $(t < MaxGenerations)$ *or* $(Stop\ Criterion)$

#Cockoo Search Algorithm

Generate new position vectors $x\ new_i$ by using CS algorithm defined in $equ\left(x_i^{t+1}\right)$

Obtain the variational parameters by multiply by subtracting position and initial position

#PSO Updation

Update the villosity and position using v_i^{t+1} and a_i^{t+1} with addition of variations in it

Calculate the fitness value E_{new_i}

Obtain g_{best}

If $(E_{newi} > E_i)$

$E_i = E\_new_i$

$p_{best_i} = anew_i$

End If

Analysis and acquire the optimum g_{best}

(Continued)

Algorithm 1 (continued)

End While
Selected features based on the optimal parameter
End

3.4 Classification of Intrusion Identification Using a Gaussian Naïve Bayes Random Forest Algorithm

The extension of Naïve Bayes known as Gaussian Naïve Bayes utilizes different functions to estimate data distributions. The Gaussian function is favored for its simplicity, requiring only the estimation of the mean and standard deviation of the training data. Consequently, the Gaussian density function is employed in continuous data classification to determine the probability of each operational class. To achieve optimal classification of selected data, classifier techniques have been hybridized with both random forest and Gaussian Naïve Bayes algorithms. The pseudocode for the Gaussian Naïve Bayes Random Forest classifier is provided in Algorithm 2. Similar to the aforementioned algorithm, the proposed methodology also exhibits a time complexity of $(O(n*d*k))/(O(d*c))$ and a space complexity of $O(depth*k)$, where n denotes the number of data points, d represents search dimensions, k indicates the number of nearest neighbors, and c signifies the number of classes. On average, this algorithm delivers an optimal solution for classifying data from selected features within 1.9 milliseconds. Algorithm 2 outlines the Gaussian Naïve Bayes algorithm in detail.

Algorithm 2: Gaussian Naïve Bayes algorithm

Input
Training dataset TS
$F = (feat_1, feat_2, feat_3, .., feat_n)$ in the testing dataset
Output
A class of testing dataset
Step
1. Read the training dataset.
2. Calculate the mean and standard deviation for the predictor variables in each class.
3. Repeat
4. Calculate the probability of $feat_i$ using the guess density equation in each class until the probability of all predictor variables $(feat_1, feat_2, feat_3, \ldots, feat_n)$ has been calculated.
5. Calculate the likelihood for each class using Hi matrices from the Random Forest features and combine with the mean of Gaussian likelihood values.
6. Get the greatest likelihood.

4 Results and Discussion

4.1 Dataset Description

The NSL-KDD dataset has been used to validate the performance of the technique which is implemented in this research and was obtained from https://www.kaggle.com/hassan06/nslkdd (accessed on 13 June 2024). The number of records in both the training and testing datasets is carefully considered to ensure affordability when executing experiments without the need to randomly select a small subset. Notably, the training set of this dataset is devoid of redundant records, ensuring that the classifiers do not yield biased results. Furthermore, the test dataset does not contain duplicate records,

which contributes to a better reduction rate [27]. The acquired dataset was partitioned and 80% of the data was given for training purposes and reserving the remaining 20% for testing.

4.2 Feature Extraction

The proposed model combines or hybridizes two methodologies: the Gated Recurrent Unit with the Autoencoder technique for feature extraction. Prior to feature extraction from the dataset, there are 45 available features. However, after feature extraction, the total number of features is extended to 128. This results in an increment in extracted features by 83 from the dataset using the suggested feature extraction model. The proposed feature extraction model enhances performance by providing filters that may better fit the data.

4.3 Hyper Parameter Optimization

During the implementation of feature extraction, a neural network integrated with the autoencoder was utilized. An essential aspect of neural network design lies in the selection of activation functions, as they significantly influence the attainable outcomes [28]. The optimal network configuration was determined by conducting a search across the hyperparameter space. These parameters, which include the optimizer, batch size, epochs, and dense layer, along with their respective accuracies, were noted. The aforementioned hyperparameters were documented for the GRU-Autoencoder neural network, and their corresponding accuracy rates are presented in Table 1.

Table 1: Hyper parameters in the GRU-auto encoder model

Activation	Batch size	Optimizer	Epochs	Dense layer	Accuracy
tanh	10	adam	100	50	0.954512
tanh	10	rmsprop	100	10	0.96874
tanh	10	adam	100	50	0.954512
relu	10	rmsprop	100	100	0.961243
relu	10	SGD	100	120	0.961876
tanh	10	rmsprop	100	10	0.96874
relu	10	adam	30	100	0.968539
relu	10	SGD	30	128	0.9982

As per the above table, the obtained result displayed that the maximum accuracy rate resulted with the 99.8% while using the activation layer as relu and optimizer as SGD when compared with the other models.

4.4 Feature Selection

In parallel with feature extraction, feature selection was devised by integrating two distinct techniques: variational cuckoo search and the PSO algorithm. From the initial pool of 128 features extracted during the feature extraction process, the proposed feature selection methods further refine the selection, ultimately identifying 20 features for subsequent processing. Table 2 delineates the chosen features derived from the previously extracted feature set.

Table 2: Number of features extracted using the proposed feature extraction model

Total number of columns	Number of selected columns
128	2, 3, 5, 6, 8, 120, 10, 12, 16, 89, 20, 98, 29, 82, 56, 34, 36, 47, 42, 43

Prior to the feature selection process, the accuracy attained was 97.56%, while post feature selection, the accuracy increased to approximately 99.82%. This indicates a 2.26% increase in accuracy resulting from the feature selection procedure. The proposed feature selection technique demonstrates strong predictive performance, yielding the highest accuracy, as depicted above in Fig. 4.

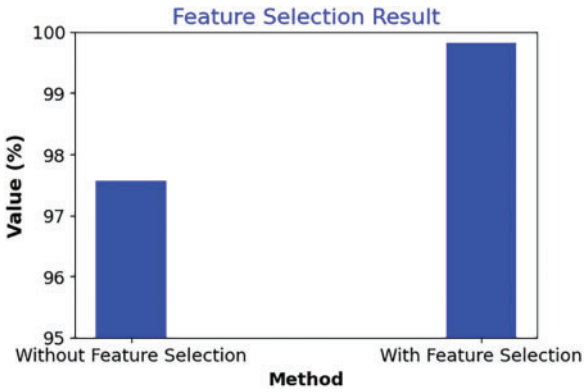

Figure 4: Accuracy rate for the proposed feature selection technique

4.5 Performance Matrices

To demonstrate the performance of the proposed methodology, various performance metrics are described below [29,30]:

Accuracy: Accuracy measures the amount of the entire correct classifications number.

$$Accuracy\ rate = \frac{TP + TN}{TP + FN + TN + FP} \tag{11}$$

Precision: Precision is the process of measuring the number of precise classifications with the inappropriate classification.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

Recall: Recall is the process of measuring the precise classification with the wasted entries.

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

F1-score performance: The measure of F1-score helps to find the mean harmonic of the recall and precision.

$$F1 - score = \frac{Precision * Recall}{Precision + Recall} \tag{14}$$

where *TP* denotes the samples that correctly classified as an attack class, *TN* denotes the samples correctly classified in normal class, *FP* denotes the total samples that have incorrectly classified class attack as well as *FN* denotes the total samples classified correctly in the normal class.

4.5.1 Accuracy Rate

This section provides an extensive evaluation and comparison with other existing methodologies, including XGBoost-DNN, LR-Logistic Regression, NB-Naïve Bayes, and SVM. Fig. 5 presents a comparison of the accuracy rates between the proposed model and the existing models for intrusion detection in IoT networks. The figure illustrates that the accuracy rate is highest in the proposed methodology compared to existing methodologies, reaching approximately 99.81% for the intrusion detection system.

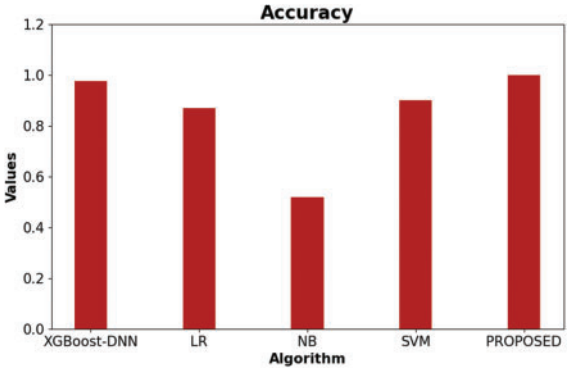

Figure 5: Evaluation result based on accuracy of proposed from existing methodology

4.5.2 Evaluation Based on Precision

A comprehensive evaluation of precision metrics for the proposed methodology compared to existing methods is presented in this section. Fig. 6 below illustrates the precision rates of the proposed methodology alongside previously presented methods for detecting intrusion in IoT network traffic. The figure demonstrates that the proposed methodology achieves a maximum precision rate of 99.1%, surpassing that of other existing methodologies implemented in intrusion detection systems.

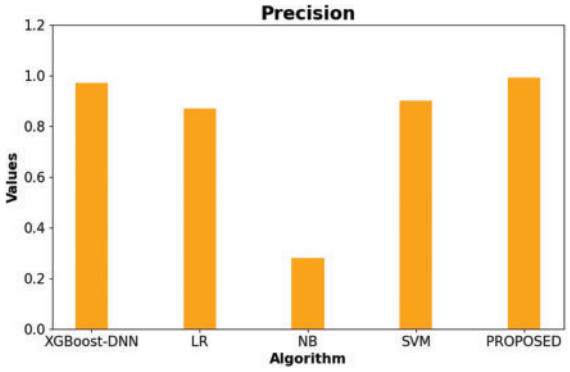

Figure 6: Evaluation result based on precision

4.5.3 Evaluation Based on Recall

A detailed evaluation is presented in this section that shows the results for the recall metrics of the developed model compared to existing models across various types of IDSs. Fig. 7 illustrates the recall rates for the developed model in conjunction with existing models. The figure below shows a maximum recall rate of 98.75%, surpassing that of other existing methodologies for intrusion detection in IoT environments.

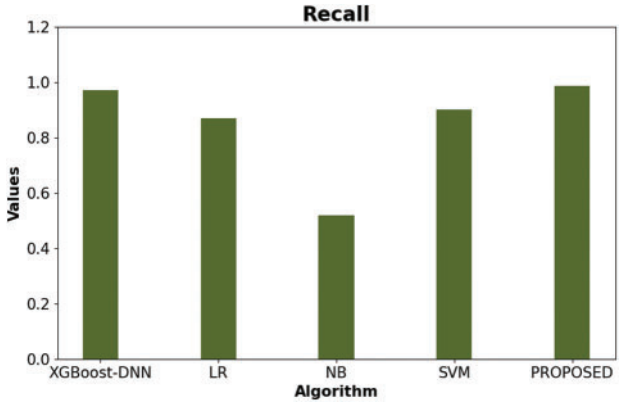

Figure 7: Evaluation result based on recall

4.5.4 Evaluation Based on F1-Score Metrics

This section provides a comprehensive evaluation of the F1-score metrics for the implemented methodology compared to existing methodologies. Fig. 8 illustrates the evaluation of the F1-score metrics for the proposed methodology alongside existing methodologies. The figure below demonstrates that the proposed methodology achieves a maximum F1-score of 99.45%, surpassing that of other existing methodologies.

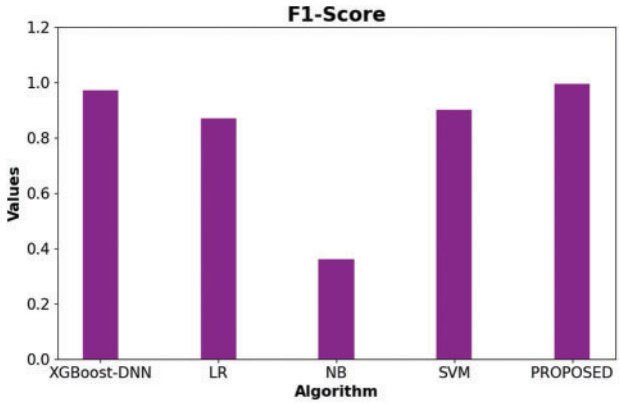

Figure 8: Evaluation results based on F1-score metrics

4.6 Comparative Analysis

A summarized comparison of the results between the proposed methodology and conventional methodologies is provided in Table 3, highlighting the superior accuracy and detection rate achieved

by the proposed approach. Fig. 9 illustrates the analysis of execution time between the proposed methodology and other existing methodologies. For instance, the GWO-SVM-Intrusion Detection System with five wolves and seven consumes approximately 74.4 and 69.6 h, respectively. Similarly, GWO-SVM-IDS with three wolves and PSO-IDS require processing times of 86.4 and 129.6 h, respectively, for detecting all intrusions. On the contrary, the proposed methodology achieves an execution time of approximately 32.56 h for intrusion detection, indicating a significant reduction in processing time required by the IDS.

Table 3: Results summary of proposed methodology with existing methods

Technique	Detection rate	Accuracy	Computation time	Number of features
GWO-SVM-IDS -seven wolves	0.96	0.96	69.6	12
GWO-SVM-IDS 3 wolves	0.83	0.79	86.4	27
GWO-SVM-IDS- 5 wolves	0.96	0.92	74.4	12
PSOSVM-IDS	0.93	0.89	129.6	20
Proposed	0.998	0.9982	32.56	20

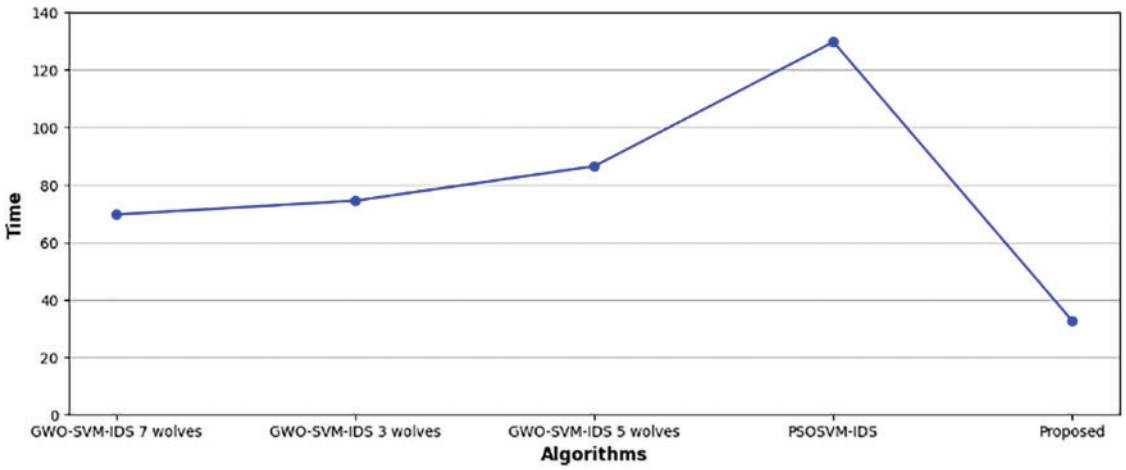

Figure 9: Comparative analysis based on execution time

The analytical findings indicate that the proposed system is effective and more efficient than conventional methods based on the considered metrics [31,32]. The accuracy rate and execution time of the introduced methodology are minimal, which demonstrates its effectiveness. These outcomes render it well-suited for intrusion detection purposes.

5 Conclusion

The IDS system functions by promptly monitoring IoT network traffic to detect threats, attacks, or any suspicious activities. Therefore, this study employs the integration of autoencoder and GRU algorithms for feature extraction as its primary stage. This amalgamation of distinct techniques for feature extraction aids in distinguishing between normal and abnormal events. Following feature extraction, feature selection is implemented using a combination of CS and PSO techniques to select appropriate features from the NSL-KDD datasets. Subsequently, after feature selection, a suitable

classification is performed using the proposed classifiers by integrating Random Forest with Gaussian Naïve Bayes classification. Typically, Naïve Bayes technique is employed for predicting test dataset classes efficiently. By extending Naïve Bayes to Gaussian Naïve Bayes, the process is simplified, as it only requires estimating the standard deviation and mean of the given training dataset. Moreover, computation time is analyzed for each algorithm used in this model. In addition, various performance matrices were used to evaluate the proposed method with different existing models such as XGBOOST-DNN, Logistic Regression, and Naïve Bayes. The computation time for each algorithm is analyzed, revealing a reduced computation time compared to other existing techniques. The proposed model achieves a maximum accuracy rate of 0.9981 for the NSL-KDD dataset, while the existing model provides the highest accuracy rate of 0.97. Furthermore, the detection rate demonstrates that the proposed methodology outperforms existing techniques in classifying and detecting various intrusions in the IoT environment. Thus, the system effectively restores minimal time. This work could be extended by utilizing different datasets and transfer functions for feature extraction to further improve model performance in the future.

Acknowledgement: The authors thank the Department of Computer Science and the Deanship of Scientific Research at Shaqra University for providing the necessary infrastructure and facilities to conduct this research.

Funding Statement: The authors extend their appreciation to the Deanship of Scientific Research at Shaqra University for funding this research work through the project number (SU-ANN-2023051).

Author Contributions: The authors confirm the contribution to the paper as follows: study conception and design: Mohammad Shoab, Loiy Alsbatin; data collection: Mohammad Shoab, Loiy Alsbatin; analysis and interpretation of results: Mohammad Shoab, Loiy Alsbatin; draft manuscript preparation: Mohammad Shoab, Loiy Alsbatin. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Any data and material presented in this study is available within the article.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] 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.

[2] 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. Telecom. Tech.*, vol. 32, 2021, Art. no. e4150. doi: 10.1002/ett.4150.

[3] T. Saranya, S. Sridevi, C. Deisy, T. D. Chung, and M. K. A. A. Khan, "Performance analysis of machine learning algorithms in intrusion detection system: A review," *Procedia Comput. Sci.*, vol. 171, no. 4, pp. 1251–1260, 2020. doi: 10.1016/j.procs.2020.04.133.

[4] S. Fenanir, F. Semchedine, and A. A. Baadache, "Machine learning-based lightweight intrusion detection system for the internet of things," *Rev. Intell. Artif.*, vol. 33, no. 3, pp. 203–211, 2019. doi: 10.18280/ria.330306.
[5] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41238–41248, 2018. doi: 10.1109/ACCESS.2018.2858277.
[6] S. Kim, L. Chen, and J. Kim, "Intrusion prediction using LSTM and GRU with UNSW-NB15," in *2021 Comp., Comm. IoT Appl. (ComComAp)*, Shenzhen, China, Nov. 2021, pp. 101–106.
[7] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, "Network intrusion detection based on supervised adversarial variational auto-encoder with regularization," *IEEE Access*, vol. 8, pp. 42169–42184, 2020. doi: 10.1109/ACCESS.2020.2977007.
[8] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert Syst. Appl.*, vol. 148, no. 13, 2020, Art. no. 113249. doi: 10.1016/j.eswa.2020.113249.
[9] S. Kulshrestha and A. Goyal, "Cuckoo search algorithm and BF Tree used for anomaly detection in data mining," *Int. J. Comp. Org. Trends*, vol. 9, no. 4, pp. 11–18, 2019. doi: 10.14445/22492593/IJCOT-V9I4P303.
[10] P. K. Keserwani, M. C. Govil, E. S. Pilli, and P. Govil, "A smart anomaly-based intrusion detection system for the internet of things (IoT) network using GWO–PSO–RF model," *J. Reliab. Intell. Environ.*, vol. 7, no. 1, pp. 3–21, 2021. doi: 10.1007/s40860-020-00126-x.
[11] M. Ajdani and H. Ghaffary, "Introduced a new method for enhancement of intrusion detection with random forest and PSO algorithm," *Secur. Priv.*, vol. 4, no. 2, pp. e147, 2021. doi: 10.1002/spy2.147.
[12] A. Mehmood, M. Mukherjee, S. H. Ahmed, H. Song, and K. M. Malik, "NBC-MAIDS: Naïve Bayesian classification technique in multi-agent system-enriched IDS for securing IoT against DDoS attacks," *J. Supercomput.*, vol. 74, no. 10, pp. 5156–5170, 2018. doi: 10.1007/s11227-018-2413-7.
[13] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building auto-encoder intrusion detection system based on random forest feature selection," *Comput. Secur.*, vol. 95, no. 1, 2020, Art. no. 101851. doi: 10.1016/j.cose.2020.101851.
[14] 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. 75729–75740, 2021. doi: 10.1109/ACCESS.2021.3082147.
[15] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82512–82521, 2019. doi: 10.1109/ACCESS.2019.2923640.
[16] 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.
[17] M. S. Ansari, V. Bartoš, and B. Lee, "GRU-based deep learning approach for network intrusion alert prediction," *Future Gener. Comput. Syst.*, vol. 128, no. 4, pp. 235–247, 2022. doi: 10.1016/j.future.2021.09.040.
[18] J. Ghosh, D. Kumar, and R. Tripathi, "Features extraction for network intrusion detection using genetic algorithm (GA)," *Modern Appr. Mach. Learn. Cogn. Sci.: A Walkthrough*, vol. 885, pp. 13–25, 2020. doi: 10.1007/978-3-030-38445-6_2.
[19] R. O. Ogundokun, J. B. Awotunde, P. Sadiku, E. A. Adeniyi, M. Abiodun and I. O. Dauda, "An enhanced intrusion detection system using particle swarm optimization feature extraction technique," *Procedia Comput. Sci.*, vol. 193, no. 1, pp. 504–512, 2021. doi: 10.1016/j.procs.2021.10.052.
[20] N. Kunhare, R. Tiwari, and J. Dhar, "Particle swarm optimization and feature selection for intrusion detection system," *Sādhanā*, vol. 45, no. 1, 2020, Art. no. 109. doi: 10.1007/s12046-020-1308-5.
[21] S. Sarvari, N. F. Mohd Sani, Z. Mohd Hanapi, and M. T. Abdullah, "An efficient anomaly intrusion detection method with feature selection and evolutionary neural network," *IEEE Access*, vol. 8, pp. 70651–70663, 2020. doi: 10.1109/ACCESS.2020.2986217.
[22] W. He, Y. Liu, H. Yao, T. Mai, N. Zhang and F. R. Yu, "Distributed variational bayes-based in-network security for the internet of things," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6293–6304, 2021. doi: 10.1109/JIOT.2020.3041656.

[23] N. B. Nanda and A. Parikh, "Network intrusion detection system based experimental study of combined classifiers using random forest classifiers for feature selection," *Int. J. Res. Electron Comput. Eng.*, vol. 6, no. 4, pp. 341–345, 2018.

[24] T. Nathiya and G. Suseendran, "An effective way of cloud intrusion detection system using decision tree, support vector machine and Naïve Bayes algorithm," *Int. J. Recent Technol. Eng.*, vol. 7, pp. 38–42, 2018.

[25] N. Ding, H. Ma, H. Gao, Y. Ma, and G. Tan, "Real-time anomaly detection based on long short-term memory and Gaussian Mixture Model," *Comput. Electr. Eng.*, vol. 79, no. 1, 2019, Art. no. 106458. doi: 10.1016/j.compeleceng.2019.106458.

[26] A. Thakkar and R. Lohiya, "A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges," *Arch. Comput. Methods Eng.*, vol. 28, no. 4, pp. 3211–3243, 2021. doi: 10.1007/s11831-020-09496-0.

[27] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020. doi: 10.1109/ACCESS.2020.2972627.

[28] M. Choraś and M. Pawlicki, "Intrusion detection approach based on optimised artificial neural network," *Neurocomputing*, vol. 452, no. 5, pp. 705–715, 2021. doi: 10.1016/j.neucom.2020.07.138.

[29] Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: A deep learning-based intrusion detection framework for securing IoT," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 3, 2022, Art. no. e3803. doi: 10.1002/ett.3803.

[30] P. Devan and N. Khare, "An efficient XGBoost–DNN-based classification model for network intrusion detection system," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12499–12514, 2020. doi: 10.1007/s00521-020-04708-x.

[31] M. Safaldin, M. Otair, and L. Abualigah, "Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 2, pp. 1559–1576, 2021. doi: 10.1007/s12652-020-02228-z.

[32] S. Saif, N. Yasmin, and S. Biswas, "Feature engineering based performance analysis of ML and DL algorithms for Botnet attack detection in IoMT," *Int. J. Syts. Assur. Eng. Manag.*, vol. 14, no. S1, pp. 512–522, 2023. doi: 10.1007/s13198-023-01883-7.