Bulletin of Electrical Engineering and Informatics
Vol. 13, No. 5, October 2024, pp. 3616~3627
ISSN: 2302-9285, DOI: 10.11591/eei.v13i5.7548

Hybridized grasshopper optimization and cuckoo search algorithm for the classification of malware

Chandini Shivaramu Banumathi, Ajjipura Basavegowda Rajendra
Department of Information Science, Vidyavardhaka College of Engineering, Mysuru, Affiliated to Visvesvaraya Technological University, Belagavi, India

Article Info

Article history:

Received Sep 12, 2023
Revised Mar 16, 2024
Accepted Mar 31, 2024

Keywords:

Cuckoo search algorithm
Grasshopper optimization algorithm
Long-short term memory
Malware classification
Softsign function

ABSTRACT

The classification and analysis of malicious software (malware) has reached a huge development in the systems associated with the internet. The malware exploits the system information and takes off the important information of the user without any intimation. Moreover, the malware furtively directs that information to the servers which are organized by the attackers. In recent years, many researchers and scientists discovered anti-malware products to identify known malware. But these methods are not robust to detect obfuscated and packed malware. To overcome these problems, the hybridized grasshopper optimization and cuckoo search (GOA-CSA) algorithm is proposed. The effective features are selected by the GOA-CSA algorithm which eases the process of classifying the malware. This research also utilized long short-term memory (LSTM)-softsign classifier to classify the malware. The malware samples are collected from the VXHeavens dataset which consists of malware samples from various software. The proposed model performance is estimated by using the performance metrics like accuracy, sensitivity, recall, and F1-score. The model attained better accuracy of 98.95% when the model is compared with other existing models.

Corresponding Author:

Chandini Shivaramu Banumathi
Department of Information Science, Vidyavardhaka College of Engineering, Mysuru
Affiliated to Visvesvaraya Technological University
Belagavi-590018, India
Email: tchannie@gmail.com

1. INTRODUCTION

Malware is the shortest form of malicious software and is format of either code or file that delivers over a network and which steals, infects, explores, or virtually conducts the behavior attackers needed [1]. Malware is software that is particularly created for damaging, disrupting, or gaining unnecessary access to computer devices [2]. Worldwide, malware attacks are the main issue that cause serious impacts on cyberspace. The anti-malware firms and the researchers put their effort into detecting the malware attacks and there is a constant improvement in malware which is noted by a current symantec cybersecurity threat report [3]. Malicious apps primarily infiltrate a marketplace run by third parties, including the widely popular Google Android Market, which cannot guarantee the absence of problematic apps [4]. Android malware poses numerous significant security concerns for users [5]. Various malware detecting techniques are developed to protect android devices. In recent times, the signature-based technique is one the major widely used detecting techniques [6]. The signature-based technique requires experts for defining the malware signatures manually and the malware is identified by utilizing searching for signatures in apps [7]. A significant drawback of the signature-based approach is its reliance on obfuscation techniques. Malware authors can modify malware signatures, allowing the malware to evade detection by the scanning engine [8].

To overcome the above issue, an intelligent malware-detecting technique is developed. By utilizing the machine learning (ML) technique, the intelligent malware detecting technique detects the malware [9]. The hidden patterns of malware are learned by ML techniques. The hidden patterns contain a high generalization capacity that differentiates malicious apps from benign ones [10]. Commonly, traditional ML detecting techniques requires dynamic monitoring or decompiling techniques for analyzing the malware and the feature representation technique is developed for re-representing malware [11], [12]. The above-mentioned technique is time-consuming and mainly depends on the expert's knowledge and the technique is processed in a monitoring environment because of its hiding behavior at the time it detects [13]. In those cases, it is difficult to observe the malware feature representation [14]. ML-based techniques can use searching knowledge and deep learning-based techniques can learn the difficult patterns of malware which show some improved outputs in the detection of obfuscated malware. In previous research, API calls are mainly utilized for feature extraction [15]. The malware detection system for various attacks majorly depended on signature-based malware detection technique, that compared the incoming traffic to saved signature of previously known malware to find potential attacks [16]. When these approaches are exact and efficient in detecting the known attacks, but failed to detect new or unknown attacks. Furthermore, they require relevant resources and manual intervention to update the attacks and as an outcome they can't detected the zero-day and unknown attacks [17]. The previous research utilized stepwise regression and CV to overcome the malware problems, but the outcomes of these methodologies don't provide enough classification results [18].

Xing *et al*. [19] presented a neural network technique that depends on auto encoder networks for the detection of malware using the Google App Store in benign and virus share. The presented model produces feature images for every benign and malware software which converts different methods bytecodes into grey-scale images for the classification and training of the model. The model extracted the key features better and guarantee the test results' accuracy. The dataset utilized for the classification was small and required improvement in the reduction of redundant data in the pre-processing stage. Alzaylaee *et al*. [20] introduced a deep learning (DL)-based dynamic analysis known as DL–droid for malware detection in Android by using android applications. To process the Android apps and for extracting their features an automation platform is required. The extracted features were given as input to DL droid's deep learning model for detecting the malware in Android based on classification. The introduced model shows better performance by utilizing the features observed from the input produces technique. Mahindru and Sangal [21] developed a web-based ML droid framework for detecting malware in Android devices using Android apps in the format of APK files. The developed model utilizes the trained models which were developed by ML techniques to identify the malware attacked apps. With the smaller number of known samples, the developed ML droid model produces a good detection rate. The developed model detects only the affected malware apps and not detects the particular feature which can identify the malware. Lu *et al*. [22] implemented a combined deep learning model that has a deep belief network (DBN) and gate recurrent unit (GRU) for the detection of Malware by using samples of Android malware and benign. The feature vectors of dynamic and static vectors were utilized for training the DBN and GRU deep learning models. The result of vectors was given as input to fully connected layer and activation function softmax identifies result of many neurons and final classification result was the format of probability. The major benefits of using this model were learning speed of the static features for Android apps was faster. The model detected malware better in known samples and for unknown samples it could not be detected efficiently. Hemalatha *et al*. [23] introduced a deep learning model based on Dense Net for the efficient detection of malware in binary images. By giving input images into the initial convolution layer, the model was trained and the convolutional neural network (CNN) has a high ability for extracting the features. The model trained the DenseNet and utilized for extracting the features from malware datasets. The introduced deep learning model for malware detection improves a high accuracy rate without the complex feature engineering tasks. Alhogail and Al-Turaiki [24] introduced a gradient boosted machines for malicious detection using DNS dataset. By utilizing the GBM classification with integration of host and lexical based features generated the much accurate detection rates. The method failed to detect the unknown attacks. Stiawan *et al*. [25] implemented a k-nearest neighbors (KNN) method and CFG-4-gram for the ransomeware detection using VXHeavens datatset. The implemented algorithm was a type of supervised learning utilized to detect unknown files and classified in accordance with pre-existing classes. There was no initialization to detect the ransomeware in beginning phases. Azeez *et al*. [26] utilized one-dimensional convolutional neural network (1D-CNN) method for the malware detection in PE files acquired from Kaggle. The utilized method integrated with ExtraTrees classifier as meta-learner for the detection of malware. The Utilized attained the high detection rate, however certain malicious codes can't be identified.

Damaševičius *et al*. [27] presented an ensemble-based classification by utilizing ML and neural networks models for detecting the malware functions in Windows PE. At the initial stage, the stacked ensemble of dense called fully connected and CNN classification was performed and classification utilizing meta-learner was performed at the final stage. The model showed that using lightweight deep learning

models based on an ensemble learning framework successfully solves the detection of malware problems. The dataset used in the model was smaller and for larger datasets how the model performs will not identified. Naeem *et al*. [28] implemented a combined deep learning and image visualization model based on the malware detection in industrial internet of things (MD-IIoT) model for malware detection. A combined technique mainly concentrated on a deep convolution neural network and color image visualization was implemented for analyzing malware in-depth. The deep CNN model was cost-effective, had scalable computation, and low run-time costs. The traditional detecting techniques of malware were not directly used for IIOT devices because of their complex computing and constrained resources like memory, cost, and constrained processing ability. To improve the classification accuracy in malware analysis, the relevant features are required to be chosen which reduces the redundant and irrelevant feature, that can improve the detection rate and reduces the false positive rate. From the existing methods, utilized various single optimization and hybrid optimization algorithms for feature selection. However, those algorithm's performance and convergence were affected by various factors. As well as the balance between exploration and search space has main cause on algorithm convergence. Among them, algorithm which has most exploration times can't identify global solution and may fall into local optimum. In this research, for feature selection, the hybrid grasshopper optimization algorithm (GOA) with cuckoo search algorithm (CSA) is proposed. The GOA has limitations like slow convergence speed and easily fall into local optimum. To overcome these limitation GOA ia combined with CSA which is effeicint in global search and balance between local and global search. The long short-term memory (LSTM) is utilized for the classification of malware which is easily influence to the problem of vanishing gradient. To overcome this problem an activation function is required, here Softsign activation function is utilized which conserved data for long sequences that enhances the classification accuracy. The main contributions of the research are listed:

- This research proposed a hybrid grasshopper optimization and cuckoo search (GOA-CSA) algorithm for feature selection, which balance between local and global search and selects the relevant features for effective classification.
- The LSTM-Softsign classifier utilized in this research maps with a large probability value and obtains better classification performance.
- The performance of the proposed method is estimated with performance metrics like accuracy, sensitivity, recall, F1-score, AUC and detection rate.

The rest of manuscript is categorized as follows, section 2 gives details of proposed method. The Grasshopper and cuckoo Search optimization algorithm is described in section 3. The results and discussion are described in section 4. Finally, section 5 describes conclusion of manuscript.

2. PROPOSED METHOD

The classification of malware in software codes and software applications is not an easy task in the interpretation of malware classification. This research proposed a hybrid of the GOA-CSA algorithm which is utilized in feature selection. The features selected using the hybrid GOA-CSA make the classification process easier by providing better accuracy and classifying the malware from the software applications. The overall process involved in the classification process is diagrammatically represented in Figure 1.

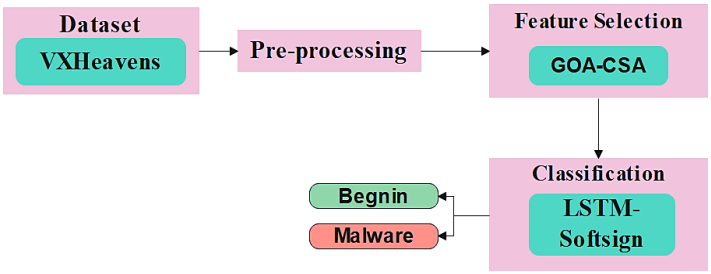

Figure 1. Represents the flow of the proposed method

2.1. Dataset

This research collected malware samples from the dataset known as VXHeavens [29] which consists of 2129 sample files of which 932 are malware samples and the benign software samples are gathered from Windows XP system directory and 346 PE format EXE files such as multimedia software and graphics

software. The malware samples from benign software are identified from 265 anti-virus software applications. The VXHeavens website has 586 samples which include viruses, worms, trojans, and ransomware. The dataset is analyzed in the malicious total virus website and the malicious running process screenshot is attached as Figure 2 and Table 1 represents the description of dataset.

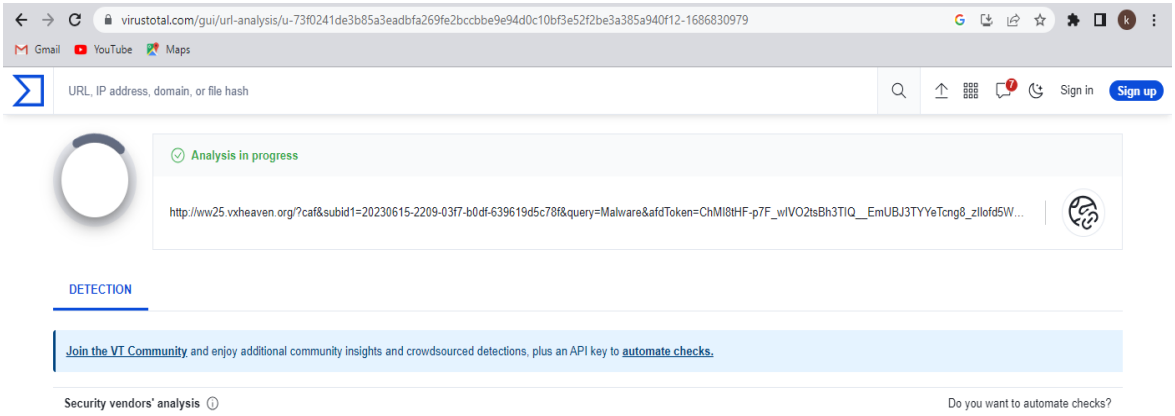

Figure 2. Malicious running process

Table 1. Dataset description

Dataset	Malware	Benign	Anti-virus	Other viruses	Total
VXHeavens	932	346	265	586	2129

2.2. Pre-processing

After collecting the data from the datasets, pre-processing is necessary to be performed. Since the malware files consist of redundant data and unnecessary files they should be removed from them [30]. The malware files obtained from the VXHeavens dataset are pre-processed to select the required features from the dataset. Here, pre-processing is performed to obtain the selected features from the dataset.

3. FEATURE SELECTION

The preprocessed data is used for feature selection by utilizing hybrid GOA-CSA algorithm. Feature selection in malware analysis can reduce the redundant and irrelevant feature and improves the detection rate and reduces the false positive rate. The previous research utilized stepwise regression and CV to overcome the malware problems, but the outcomes of these methodologies don't provide enough classification results. To overcome these issues, the hybrid of GOA-CSA is introduced which selects stochastic features and provides better classification accuracy than the existing methods.

3.1. Grasshopper optimization algorithm

GOA is a population-based meta-heuristic optimization approach. The preprocessed data are given to the GOA algorithm for selecting the relevant features to be used in classification. The GOA has limitations like slow convergence speed and easily fall into local optimum. To overcome these limitation GOA ia combined with CSA which is effeicint in global search and balance between local and global search. The proposed algorithm effectively reduced the irrelevant features and provides effective performance. Generally, Grasshoppers are known as insects which are considered as most dangerous ones to crops. The major thing in grasshoppers is the need to create a swarm even though they are naturally separated. The collection of swarms by grasshoppers is the highest swarms of overall swarms in global and that is taken as nightmare to farmers. The grasshopper's lifecycle includes egg, nymph, and adult and eggs are incubated for 10 months and nymph grasshopper is born. The nymph grasshopper jumps over and started rolling and ate anything which comes on their path. After certain days the nymph grasshopper becomes an adult and starts streaming in air.

Generally, GOA concentrated on grasshopper's social characteristics. Each limb in the swarm has one insect placed in a position of 'S' search space and moves in its limit. The two significant movements of the grasshopper. The first one, grasshopper coordination moves slowly at the stage of larvae and moves dynamically at the stage of insect format. The second one, searching for food. For forming a swarm, the

grasshopper agents are randomly generated. By utilizing the evaluation of fitness value, the good search agent is selected. The good search agent started moving towards other independent with their surroundings. So, each search agent starts moving towards the good search agent of Grasshopper M_i represents the ith search agent of feature migration along the required search agent. In (1) represents M_i in feature selection,

$$M_i = S_i + G_i + A_i \tag{1}$$

A_i the above equation represents the wind direction, G_i represents the ith grasshopper gravitational force and S_i represents social interaction, that is taken as major movement state of grasshopper motion in feature selection. It is given in (2), in (2) where, $\hat{d}_{ij}$ represents the ith to jth grasshopper position vector format and d_{ij} represents the two grasshopper's distance and the mathematical format is described in (3). In (3), S represents the social interaction force and the mathematical format of social interaction force is described in (4). In (4), a represents the attractive scale length, d represents the in-between distance of grasshoppers and f represents social forces strength and the gravitational force G_i is calculated in (5),

$$S_i = \sum_{i=1}^{N} S(d_{ij})\hat{d}_{ij}\ j \neq i \tag{2}$$

$$d_{ij} = |m_j - m_i| \tag{3}$$

$$S(r) = fe^{\frac{-d}{a}} - e^{-d} \tag{4}$$

$$G_i = -g\hat{e}_g \tag{5}$$

Where g represents the gravitational force constant and $\hat{e}_g$ represents the unit vector. $A_i\ the$ component is represented as (6). In (6), v represents the drift velocity constant and $\hat{e}_v$ represents the unit vector. According to the proposed model, (7) is given as follows. In modifying the proposed model for energy optimization issues in feature selection, in (8) is utilized.

$$A_i = v\hat{e}_v \tag{6}$$

$$M_i = \sum_{j=1,j\neq1}^{N} S(|P_j - P_i|)\frac{(P_j - P_i)}{d_{ij}} - g\hat{e}_g + v\hat{e}_v \tag{7}$$

$$M_i = \gamma\left(\sum_{j=1,j\neq1}^{N} \gamma\frac{u_u - u_l}{2} S(|m_j - m_i|)\frac{(m_j - m_i)}{d_{ij}} - g\hat{e}_g + v\hat{e}_v\right) + \hat{\tau}_d \tag{8}$$

Where u_u represents the upper bounds in dimension D, u_l represents the lower bounds in dimension D, γ represents the parameter of decreasing that contract comfort zone and $\hat{\tau}_d$ represents target agent D dimension value. The GOA is utilized in this section which helps in providing better results, but it is not capable to rectify the issues regarding the selection of constraints. To overcome this issue, the research utilized CSA which is hybridized with GOA to select the best feature.

3.2. Cuckoo search algorithm

CSA is bio-inspired meta-heuristic algorithm. In explaining cuckoo search approach, each single cuckoo bird puts one egg in an individual time interval and in nest of other birds which is randomly selected host nest quantity is constant and possibility of the host searching egg is $P_h \in [0,1]$. The problem is optimized by utilizing fitness function as objective function. The eggs in the nest are considered the result of issue and new solution is dumping the cuckoo egg. The solution obtained in nest is not good the solution is substituted by the new solution. The nest has good eggs that are called the local solution which is used for reproduction. $Levy$ air tips which are named as flights and are processed to find the result to good globally and represented as (9).

$$X_i^{t+1} = X_i + \alpha \oplus Levy(\lambda), \tag{9}$$

Where X_i^{t+1} represents the output, X_i represents the present status, α represents the step or transition size. Frye and Reynolds are the recent finds $Levy$ with help of recent research of fruit-fly flight. The fruit fly examines their landscape in a sequence straight fly with an immediate turn of 90^o which is leading to $Levy - flight - style$ intermediate scale-free search pattern. Thus, the relevant features are selected by hybridized GOA-CSA algorithm and it is given to the process of classification.

3.3. Classification using LSTM soft sign activation function

The final stage is the classification of malware using the selected features. This research utilizes LSTM network with a soft sign activation function. The LSTM is easily influence to the problem of vanishing gradient. To overcome this problem an activation function is required, here softsign activation function is utilized which conserved data for long sequences that enhances the classification accuracy. LSTM is a specific type of recurrent neural network (RNN), that can learn long-distance dependence [31]. The architecture of LSTM is represented in Figure 3. Cell state is the central component of LSTM, may add or remove data from cells and permit information selectively to pass along the door mechanism to accomplish this. The forget gate, input gate, and output gate make up an LSTM. The input gate selects data which is add to cell state and forget gate selects which data to remove from cell state. The cell state can be added once these two points is established. The output gate at last determines result of network. The process of node present in LSTM is described in as (10) to (15):

$$f_t = \sigma\left(W_f.[h_{t-1}, x_t] + b_f\right) \quad (10)$$

$$i_t = \sigma\left(W_i.[h_{t-1}, x_t] + b_i\right) \quad (11)$$

$$\bar{C}_t = softsign\left(W_c.[h_{t-1}, x_t] + b_c\right) \quad (12)$$

$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t \quad (13)$$

$$o_t = \sigma\left(W_o.[h_{t-1}, x_t] + b_o\right) \quad (14)$$

$$h_t = o_t * softsign(C_t) \quad (15)$$

Where the hidden state of the prior layer is denoted as h_{t-1}, input for the current layer is represented as x_t. The weight and biased state are denoted as W and b respectively. The sigmoid function is respresented as σ and output of the forget gate is denoted as f_t. The result from the input gate is represented as i_t and intermediate temporary state is denoted as $\tilde{C}_t$. The state of the cell present in the prior layer is denoted as C_{t-1} and the state of the cell present in the next layer is denoted as C_t. The outcome from the output gate and the hidden state is denoted as o_t and h_t respectively. Computing the outcome of the input and output gate individually doesn't provide better performance so, the outcome from the input and output gate can be distinguished using the factor $1 - f_t$. This helps to improve the cell state for the next layer in the input and output gate, it is represented in (16):

$$C_t = f_t * C_{t-1} + (1 - f_t) * \bar{C}_t \quad (16)$$

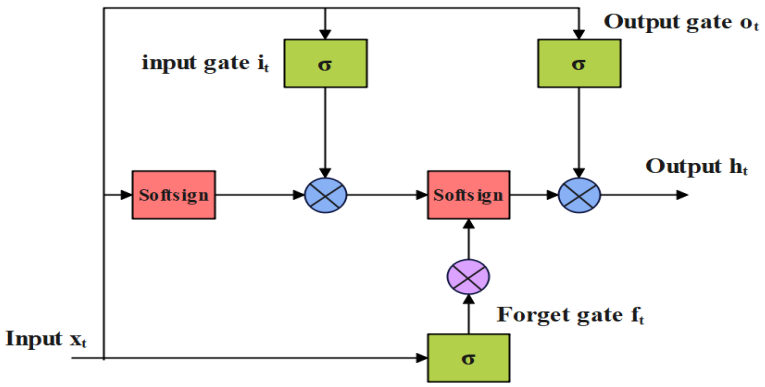

Figure 3. Structure of LSTM

3.3.1. Softsign activation function

Softsign is a type of activation function that can be utilized in LSTM. The features obtained using hybrid GOA-CSA is combined with the LSTM and the features are given to the softsign classifier in a fully connected manner to perform the classification of malware. Softsign activation function maps with a large probability value and obtain better classification performance. The softsign is a quadratic polynomial which is represented using as (17):

$$f(x) = \left(\frac{x}{|x|+1}\right) \quad (17)$$

Where the absolute input value is denoted as $|x|$. Softsign activation function maps with a large probability value and obtain better classification performance. The major variation in tanh and softsign functions is the rate of convergence, in tanh it takes place in the exponential form and softsign it takes place in the form of polynomial.

4. RESULTS AND DISCUSSION

In this part of the research, the performance of the model is estimated using the following performance metrics like accuracy, sensitivity, recall, F1-score, and AUC. The performance of proposed method is evaluated in terms of fuzzy similarity score, detection rate, fuzzy hashing detection rate, classifier performance with actual features and after selecting the features and with various feature selection methods. The mathematical representation of these parameters are given from (18) to (21):

$$Accuracy = \frac{(True\ Positive + True\ Negative)}{Total\ Instances} \quad (18)$$

$$Senstivity = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (19)$$

$$Recall = \frac{True\ Positive}{Actual\ number\ of\ Instances\ as\ True} \quad (20)$$

$$F1\ Measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (21)$$

Where TP is true positives, TN is true negatives, FP is false positives, and FN is false negatives.

4.1. Performance analysis

Table 2 represents the malware detection result of SSDEEP, SDHASH, and mvHASH-B fuzzy hashing methods for WannaCry, Locky, and Cerber Ransomware samples. The fuzzy import hashing's performance was compared to import hashing and three different hashing techniques like SSDEEP, SDHASH, and mvHASH-B. The calculation determined if this integration is successful or not, whether it is successful, then which fuzzy hashing technique generated a better accuracy result. The proposed GOA-CSA algorithm attains better performance than the other fuzzy hashing methods.

Table 2. Malware analysis of fuzzy hashing method

Fuzzy hashing matching criteria		Fuzzy similarity scores (1-100%)	Fuzzy similarity scores >10%	Fuzzy similarity scores >20%	Fuzzy similarity scores >30%
WannaCry Ransomware	SSDEEP detection rate (%)	93.5	93.5	93.5	93.1
	SDHASH detection rate (%)	95.9	95.9	92	92
	mvHASH-B detection rate (%)	92	92	86.7	86.7
	GOA-CSA detection rate (%)	94.2	94.5	93.7	93.5
Locky Ransomware	SSDEEP detection rate (%)	44	44	43.9	43.9
	SDHASH detection rate (%)	60.7	40.7	37.9	32.7
	mvHASH-B detection rate (%)	74.7	66	38.7	35.9
	GOA-CSA detection rate (%)	96.4	96.2	93.4	92.8
Cerber Ransomware	SSDEEP detection rate (%)	35.6	35.6	35.6	35.6
	SDHASH detection rate (%)	73.5	64.9	39.9	30.7
	mvHASH-B detection rate (%)	96.8	92.4	38.8	38
	GOA-CSA detection rate (%)	97.6	97.5	93.7	92.9

Table 3 represents the malware detection result of Import hashing for WannaCry, Locky and Cerber Ransomware samples. The WannaCry Ransomware obtained 89.9%, the Locky Ransomware obtained 33.9%, the Cerber Ransomware obtained 63.9% and GOA-CSA obtained 91.8% of fuzzy similarity score. The obtained result shows that the WannaCry Ransomware achieves the highest import hashing detection rate when compared to locky and cerber ransomware samples.

Table 4 represents the malware analysis result of Import hashing (IMPHASH), fuzzy hashing (SSDEEP, SDHASH, and mvHASH-B) and the fuzzy import hashing for gathered ransomware samples. The detection result of fuzzy import hashing used three various fuzzy hashing techniques for each three ransomware samples. The fuzzy similarity values higher than 30% are utilized for three fuzzy hashing

techniques. The fuzzy import hashing with three fuzzy hashing techniques shows enhancement but SSDEEP fuzzy hashing-based fuzzy import hashing gives superior outcomes.

Table 3. Malware analysis of import hashing with GOA-CSA

Classes of ransomware	Detection rate of import hashing (%)
WannaCry ransomware	89.9
Locky ransomware	33.9
Cerber ransomware	63.9

Table 4. Malware analysis of import hashing and fuzzy import hashing

Classes of ransomware	IMPHASH import hashing detection rate (%)	SSDEEP fuzzy hashing detection rate (%)	SSDEEP fuzzy-import hashing detection rate (%)	SDHASH fuzzy hashing detection rate (%)	SDHASH fuzzy-import hashing detection rate (%)	mvHASH-B fuzzy hashing detection rate (%)	mvHASH-B fuzzy-import hashing detection rate (%)	GOA-CSA detection rate (%)
WannaCry ransomware	89.9	93.1	95.2	92	95.2	86.7	95.2	96.4
Locky ransomware	33.9	43.9	51.2	32.7	47.5	35.9	46.7	50.2
Cerber ransomware	63.9	35.6	63.9	30.7	63.9	38	63.9	64.9
Overall detection rate of each hashing method	62.5	57.5	70.1	51.8	68.8	53.5	68.6	70.5

The performance of LSTM-Softsign with the existing classifiers is evaluated in this section. The analysis is performed and the results are evaluated with and without the feature selection using hybrid GOA-CSA. The existing classifiers used for evaluation are CNN, support vector machine (SVM), random forest (RF), decision tree (DT), and LSTM. The performance of the classifier without hybrid GOA-CSA (feature selector) is represented in Table 4. The performance is evaluated based on accuracy, sensitivity, recall, F-1 score and AUC.

Results from Table 5 and Figure 4, show that the LSTM-Softsign classifier attained lesser value in classification accuracy (90.81%) when compared with the existing classifiers such as DT (92.49%) and SVM (89.32%). The absence of GOA-CSA leads to reduce the performance of the LSTM-Softsign classifier. Without GOA-CSA the redundant features are directly involved in the process of classification and affect the performance of the classifier.

Table 5. Performance of classifier without feature selection

Classifier	Accuracy (%)	Sensitivity (%)	Recall (%)	F1-score (%)	AUC (%)
CNN	91.37	89.82	90.05	91.23	87.98
SVM	92.01	91.74	91.07	90.53	89.41
RF	91.97	91.52	91.01	89.32	88.76
DT	93.01	92.49	92.15	91.64	89.28
LSTM	92.59	92.03	91.45	90.15	88.73
LSTM-softsign	95.16	92.48	91.92	89.03	88.72

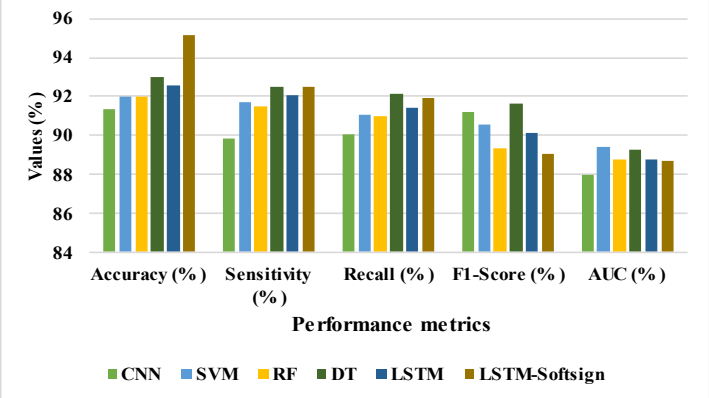

Figure 4. Graphical representation for performance of classifier without feature selection

The overall results from Table 6 and Figure 5 show that the performance of the LSTM-softsign function is higher than other classifiers by obtaining better accuracy, sensitivity, recall, and F-1 score. The combination of LSTM with soft sign function mapped the large probability values to obtain better classification performance. Moreover, the softsign classifier is designed in a fully connected way which effectively classifies the malware without any loopholes.

Table 6. Performance of classifier with feature selection

Classifier	Accuracy (%)	Sensitivity (%)	Recall (%)	F1-Score (%)	AUC (%)
CNN	91.76	90.58	90.83	91.47	88.54
SVM	92.39	92.29	91.52	90.72	89.83
RF	92.18	91.82	91.43	90.53	89.02
DT	93.64	92.68	92.53	91.73	89.63
LSTM	92.79	92.47	91.60	90.47	89.38
LSTM-softsign	98.21	97.68	97.01	95.43	89.57

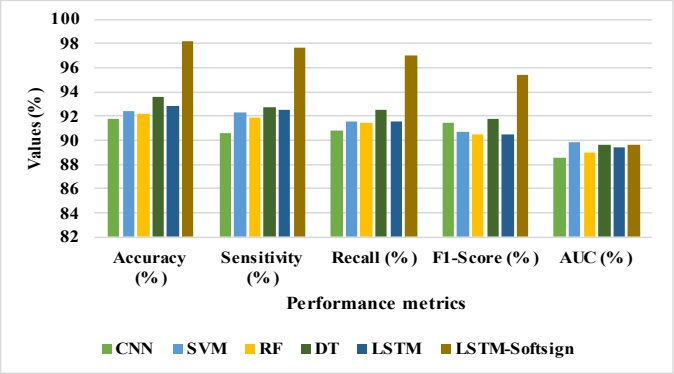

Figure 5. Graphical representation for performance of classifier with feature selection

Results from Table 7 and Figure 6 show that the hybrid of GOA-CSA achieved better performance when compared with different feature selection methodologies. GOA-CSA achieved better accuracy of 98.21% which is comparatively higher than the existing feature selection methods. The GOA-CSA achieved better results because it provides ratings to the feature subsets, ratings will be established using specific criteria, and the best subset will then be used for the creation of the model.

Table 7. Different feature selection approaches

Feature selection	Accuracy (%)	Sensitivity (%)	Recall (%)	F1-Score (%)	AUC (%)
Fisher's score	93.19	92.37	93.03	93.14	87.03
Correlation coefficient	93.63	93.28	92.50	92.05	90.18
Exhaustive feature selection	92.87	91.93	91.32	90.48	89.82
Backward feature selection	94.32	93.79	93.01	92.63	91.27
GOA-CSA	98.21	97.68	97.01	95.43	89.57

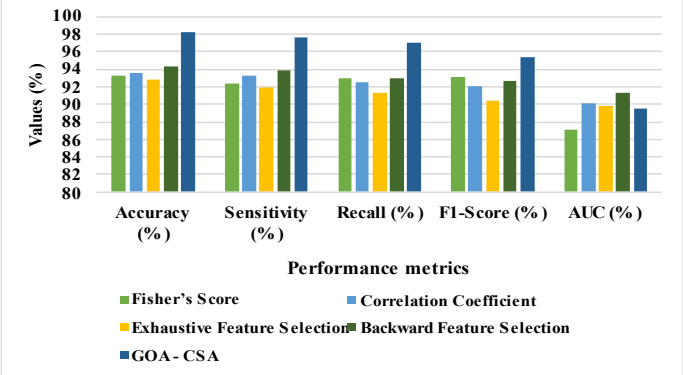

Figure 6. Performance of different feature selection approaches

4.2. Comparative analysis

The comparative analysis of proposed LSTM-Softsign classifier with the existing methods like ML-Droid [21], DBN-GRU, [22] and CNN-Softmax [23] and CFG-4-gram [25] in the classification of malware present in various software codes and applications. The comparative table of proposed LSTM-Softsign with existing classifiers is represented in Table 8.

Table 8. Comparative table

Methodologies	Accuracy (%)	Sensitivity (%)	Recall (%)	F1-Score (%)
ML-Droid [21]	98.89	95.01	93.42	92.75
DBN–GRU [22]	96.82	95.79	97.62	N/A
CNN-Softmax [23]	97.55	97.43	97.50	97.46
CFG-4-gram [25]	98.86	N/A	N/A	N/A
LSTM-Softsign	98.95	97.68	97.77	97.62

From comparative table, it is represented that proposed LSTM-Softsign classifier attained superior performance while compared to existing classifiers such as ML-Droid [21], DBN-GRU [22], CNN-Softmax [23], and CFG-4-gram [25]. The selected features from hybrid GOA-CSA are classified using the LSTM-Softsign classifier. The LSTM-softsign classifier provides better classification accuracy compared with other classifiers. This is due to the softsign layer and the performance of the LSTM-softsign function is higher than other classifiers by obtaining better accuracy, sensitivity, recall, and F1 score. The combination of LSTM with soft sign function mapped the large probability values to obtain better classification performance. The LSTM-Softsign classifier attained better classification accuracy of (98.95%) which is comparatively higher than ML-Droid (98.89%), DBN-GRU (96.82%) CNN-Softmax (97.55%), and CFG-4-gram (98.86%).

4.3. Discussion

This section illustrates the proposed method advantages and existing methods limitations. The existing model has some limitations such as ML droid [21] model detects only the affected malware apps and not detects the particular feature which can identify the malware. DBN and GRU [22] model detected malware better in known samples and for unknown samples it could not be detected efficiently. CNN-softmax [23] traditional detecting techniques of malware were not directly used for IIoT devices because of their complex computing and constrained resources like memory, cost, and constrained processing ability. The CFG-4-gram [25] method there was no initialization to detect the ransomeware in beginning phases. To improve the classification accuracy in malware analysis, the relevant features are required to be chosen which reduces the redundant and irrelevant feature, that can improve the detection rate and reduces the false positive rate. From the existing methods, utilized various single optimization and hybrid optimization algorithms for feature selection. However, those algorithm's performance and convergence were affected by various factors. As well as the balance between exploration and search space has main cause on algorithm convergence. Among them, algorithm which has most exploration times can't identify global solution and may fall into local optimum. In this research, for feature selection, the hybrid GOA with CSA is proposed. The GOA has limitations like slow convergence speed and easily fall into local optimum. To overcome these limitation GOA ia combined with CSA which is effeicint in global search and balance between local and global search. The LSTM is utilized for the classification of malware which is easily influence to the problem of vanishing gradient. To overcome this problem an activation function is required, here softsign activation function is utilized which conserved data for long sequences that enhances the classification accuracy.

5. CONCLUSION

The development of the internet and software in the modern world leads to increased threats and a lack of security due to malicious software (malware). Malware corrupts and exploits user information which may lead to a ransomware attack or other security threats. To overcome these issues, the GOA-CSA algorithm is utilized in which the effective features are selected and ease the process of classifying the malware. This research utilized the LSTM-Softsign classifier for classifying the malware and the VXHeavens dataset is utilized to obtain the malware samples. This research concludes that the proposed hybrid GOA-CSA algorithm performs well than the other existing classifiers with feature selection by providing better accuracy. The performance measures like accuracy, sensitivity, recall and F1-score are utilized for evaluating the proposed method. From the evaluation the proposed attained the high accuracy of 98.95% which is comparatively higher than other existing methods such as ML-Droid, DBN-GRU, CNN-Softmax, and CFG-4-gram. The proposed method has limitation such as it can't able to identify the zero-day attacks, limited to supervised learning, that needed both benign and malicious malware to be find, certain malicious

code can't be identified. In future, analyze on much relevant and important features of malware through experiments and integration of various feature engineering methods.

REFERENCES

[1] Y. Ding, X. Zhang, J. Hu, and W. Xu, "Android malware detection method based on bytecode image," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 5, pp. 6401–6410, May 2023, doi: 10.1007/s12652-020-02196-4.

[2] V. Ravi, T. D. Pham, and M. Alazab, "Attention-Based Multidimensional Deep Learning Approach for Cross-Architecture IoMT Malware Detection and Classification in Healthcare Cyber-Physical Systems," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 4, pp. 1597–1606, 2023, doi: 10.1109/TCSS.2022.3198123.

[3] D. A. Kumar and S. K. Das, "Machine Learning Approach for Malware Detection and Classification Using Malware Analysis Framework," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, no. 1, pp. 330–338, 2023.

[4] R. Chaganti, V. Ravi, and T. D. Pham, "Deep learning based cross architecture internet of things malware detection and classification," *Computers and Security*, vol. 120, 2022, doi: 10.1016/j.cose.2022.102779.

[5] R. N. Vaza, R. Prajapati, D. Rathod, and D. Vaghela, "Developing a novel methodology for virtual machine introspection to classify unknown malware functions," *Peer-to-Peer Networking and Applications*, vol. 15, no. 1, pp. 793–810, 2022, doi: 10.1007/s12083-021-01281-5.

[6] A. Jamal, M. F. Hayat, and M. Nasir, "Malware Detection and Classification in IoT Network using ANN," *Mehran University Research Journal of Engineering and Technology*, vol. 41, no. 1, pp. 80–91, 2022, doi: 10.22581/muet1982.2201.08.

[7] J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, "MAPAS: a practical deep learning-based android malware detection system," *International Journal of Information Security*, vol. 21, no. 4, pp. 725–738, Aug. 2022, doi: 10.1007/s10207-022-00579-6.

[8] O. A. Alzubi, J. A. Alzubi, T. M. Alzubi, and A. Singh, "Quantum Mayfly Optimization with Encoder-Decoder Driven LSTM Networks for Malware Detection and Classification Model," *Mobile Networks and Applications*, vol. 28, no. 2, pp. 795–807, 2023, doi: 10.1007/s11036-023-02105-x.

[9] F. Mercaldo, G. Ciaramella, G. Iadarola, M. Storto, F. Martinelli, and A. Santone, "Towards Explainable Quantum Machine Learning for Mobile Malware Detection and Classification †," *Applied Sciences (Switzerland)*, vol. 12, no. 23, 2022, doi: 10.3390/app122312025.

[10] A. Albakri, F. Alhayan, N. Alturki, S. Ahamed, and S. Shamsudheen, "Metaheuristics with Deep Learning Model for Cybersecurity and Android Malware Detection and Classification," *Applied Sciences (Switzerland)*, vol. 13, no. 4, 2023, doi: 10.3390/app13042172.

[11] W. T. Chew, W. V. Yong, J. S. L. Ong, J. H. Leong, and T. Sutikno, "Dynamic simulation of three-phase nine-level multilevel inverter with switching angles optimized using nature-inspired algorithm," *International Journal of Power Electronics and Drive Systems*, vol. 12, no. 1, pp. 325–333, Mar. 2021, doi: 10.11591/ijpeds.v12.i1.pp325-333.

[12] O. A. Alzubi, I. Qiqieh, and J. A. Alzubi, "Fusion of deep learning based cyberattack detection and classification model for intelligent systems," *Cluster Computing*, vol. 26, no. 2, pp. 1363–1374, 2023, doi: 10.1007/s10586-022-03686-0.

[13] P. Yadav, N. Menon, V. Ravi, S. Vishvanathan, and T. D. Pham, "A two-stage deep learning framework for image-based android malware detection and variant classification," *Computational Intelligence*, vol. 38, no. 5, pp. 1748–1771, 2022, doi: 10.1111/coin.12532.

[14] A. F. Algamluoli and N. H. Abbas, "Speed controller design for three-phase induction motor based on dynamic adjustment grasshopper optimization algorithm," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 2, pp. 1143–1157, Apr. 2021, doi: 10.11591/ijece.v11i2.pp1143-1157.

[15] W. Aribowo, B. Suprianto, and U. T. Kartini, "Cooperation search algorithm for tuning automatic voltage regulator system," *International Journal of Power Electronics and Drive Systems*, vol. 13, no. 3, pp. 1761–1769, Sep. 2022, doi: 10.11591/ijpeds.v13.i3.pp1761-1769.

[16] A. A. Alhashmi *et al.*, "Similarity-Based Hybrid Malware Detection Model Using API Calls," *Mathematics*, vol. 11, no. 13, 2023, doi: 10.3390/math11132944.

[17] A. El-Ghamry, T. Gaber, K. K. Mohammed, and A. E. Hassanien, "Optimized and Efficient Image-Based IoT Malware Detection Method," *Electronics (Switzerland)*, vol. 12, no. 3, 2023, doi: 10.3390/electronics12030708.

[18] X. W. Wu, Y. Wang, Y. Fang, and P. Jia, "Embedding vector generation based on function call graph for effective malware detection and classification," *Neural Computing and Applications*, vol. 34, no. 11, pp. 8643–8656, 2022, doi: 10.1007/s00521-021-06808-8.

[19] X. Xing, X. Jin, H. Elahi, H. Jiang, and G. Wang, "A Malware Detection Approach Using Autoencoder in Deep Learning," *IEEE Access*, vol. 10, pp. 25696–25706, 2022, doi: 10.1109/ACCESS.2022.3155695.

[20] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Computers and Security*, vol. 89, p. 101663, Feb. 2020, doi: 10.1016/j.cose.2019.101663.

[21] A. Mahindru and A. L. Sangal, "MLDroid—framework for Android malware detection using machine learning techniques," *Neural Computing and Applications*, vol. 33, no. 10, pp. 5183–5240, May 2021, doi: 10.1007/s00521-020-05309-4.

[22] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," *Security and Communication Networks*, vol. 2020, pp. 1–11, Aug. 2020, doi: 10.1155/2020/8863617.

[23] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient densenet-based deep learning model for Malware detection," *Entropy*, vol. 23, no. 3, p. 344, Mar. 2021, doi: 10.3390/e23030344.

[24] A. Alhogail and I. Al-Turaiki, "Improved Detection of Malicious Domain Names Using Gradient Boosted Machines and Feature Engineering," *Information Technology and Control*, vol. 51, no. 2, pp. 313–331, Jun. 2022, doi: 10.5755/j01.itc.51.2.30380.

[25] D. Stiawan, S. M. Daely, A. Heryanto, N. Afifah, M. Y. Idris, and R. Budiarto, "Ransomware detection based on opcode behaviour using k-nearest neighbours algorithm," *Information Technology and Control*, vol. 50, no. 3, pp. 495–506, Sep. 2021, doi: 10.5755/j01.itc.50.3.25816.

[26] N. A. Azeez, O. E. Odufuwa, S. Misra, J. Oluranti, and R. Damaševičius, "Windows PE malware detection using ensemble learning," *Informatics*, vol. 8, no. 1, p. 10, Feb. 2021, doi: 10.3390/informatics8010010.

[27] R. Damaševičius, A. Venčkauskas, J. Toldinas, and Š. Grigaliūnas, "Ensemble-based classification using neural networks and machine learning models for windows pe malware detection," *Electronics (Switzerland)*, vol. 10, no. 4, pp. 1–26, Feb. 2021, doi: 10.3390/electronics10040485.

[28] H. Naeem *et al*., "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, p. 102154, Aug. 2020, doi: 10.1016/j.adhoc.2020.102154.
[29] Link for VXHeavens dataset: http://vxheaven.org/.
[30] O. Sharma, A. Sharma, and A. Kalia, "Windows and IoT malware visualization and classification with deep CNN and Xception CNN using Markov images," *Journal of Intelligent Information Systems*, vol. 60, no. 2, pp. 349–375, 2023, doi: 10.1007/s10844-022-00734-4.
[31] D. Tian *et al*., "MDCD: A malware detection approach in cloud using deep learning," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 11, 2022, doi: 10.1002/ett.4584.

BIOGRAPHIES OF AUTHORS

Chandini Shivaramu Banumathi completed M.Tech. in Network and Internet Engineering from Sri Jayachamarajendra College of Engineering, Mysuru, Karnataka. Bachelor of Engineering in Information Science and Engineering from National Institute of Engineering, Mysuru, Karnataka. Area of interest are Information and network security, theory of computation, and machine learning. She can be contacted at email: tchannie@gmail.com.

Ajjipura Basavegowda Rajendra completed his Ph.D. in Cryptography and Network Security, M.Tech. in Master of Engineering Management from SJCE, Mysuru, Karnataka and Bachelor of Engineering in Electronics and Communication, U.B.D.T College of Engineering, under Kuvempu University in 2000. Area of interest are cryptography, network security, and internet of things. He has published articles in various journals and conferences. He can be contacted at email: rabvvce@gmail.com.