Cluster Computing (2022) 25:2369–2387
https://doi.org/10.1007/s10586-021-03459-1

An efficient malware detection approach with feature weighting based on Harris Hawks optimization

Omar A. Alzubi[1] · **Jafar A. Alzubi**[2] · **Ala' M. Al-Zoubi**[3,4] · **Mohammad A. Hassonah**[4] · **Utku Kose**[5]

Received: 16 May 2021 / Revised: 1 September 2021 / Accepted: 19 October 2021 / Published online: 8 November 2021

Abstract
This paper introduces and tests a novel machine learning approach to detect Android malware. The proposed approach is composed of Support Vector Machine (SVM) classifier and Harris Hawks Optimization (HHO) algorithm. More specifically, the role of HHO algorithm is to optimize SVM classifier hyperparameters while the SVM performs the classification of malware based on the best-chosen model, as well as producing the optimal solution for weighting the features. The effectiveness of the proposed approach and the ability to increase detection performance are demonstrated by scientific testing using CICMalAnal2017 sampled datasets. We test our method and its robustness on five sampled datasets and achieved the best results in most datasets and measures when compared with other approaches. We also illustrate the ability of the proposed approach to measure the significance of each feature. In addition, we provide deep analysis of possible relationships between weighted features and the type of malware attack. The results show that the proposed approach outperforms the other metaheuristic algorithms and state-of-art classifiers.

Keywords Machine learning · Security · Android malware detection · Harris Hawks optimization · Support vector machine · Feature weighting

1 Introduction

Over the last decade, smartphones have become one of the most used devices worldwide with nearly 3.6 billion users in 2020 according to Statista report [1]. This took place due to the outstanding functionality and features that smartphones possess [2]. Smartphones can be applied to do various features, such as sending emails, gaming, take pictures and video recording, information search, GPS, and so on. This can be done because of the applications or apps created and upgraded on a daily basis particularly, on an android operating system (OS).

Android OS was designed in 2007 as a modified version of the Linux kernel for touchscreen mobile devices. The Andriod OS obtain 70% of the market share worldwide compared to other OS in 2021 [3]. Furthermore, the number of apps on Android OS reached nearly 2.7 million apps last year [4]. These android apps can be employed in diverse categories, including banking, social media, medical, educational, and entertainment apps [5]. Consequently, most of these apps used for the sake of users' advantage. However, some of them are utilized for

✉ Omar A. Alzubi
o.zubi@bau.edu.jo

Jafar A. Alzubi
j.zubi@bau.edu.jo

Utku Kose
utkukose@sdu.edu.tr

1 Prince Abdullah bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University, Al-Salt, Jordan

2 Faculty of Engineering, Al-Balqa Applied University, Al-Salt, Jordan

3 School of Science, Technology and Engineering, University of Granada, Granada, Spain

4 Prince Abdullah bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University, Al-Salt, Jordan

5 Department of Computer Engineering, Suleyman Demirel University, Isparta, Turkey

malicious purposes in order to hack or exploit. Such malicious apps are known as Malware, which can be defined as intrusive software that steal information and damage other users devices [6, 7]. Malware is usually developed by cybercriminals and programmed to act like worms, adware, Trojan viruses, ransomware, and spyware [8].

In view of the rapid rate of development and existence of malware apps, it becomes hard to prevent and stop most security attacks [9]. These attacks are able to operate in different cases, for example, in 2019 a warning took place by Cybersecurity Check Point for android users that more than 25 million mobile devices were exposed to malware known as Agent Smith [10]. Moreover, the malware disguises under apps like WhatsApp to exploit Android OS vulnerabilities. Another example coccus in 2020 stated that above billion android devices endanger of getting hacked due to lacking new security updates [11]. These exposed devices by ransomware are the ones released before 2012. Furthermore, according to Kaspersky Lab researchers in 2020, a number of hackers have been employing Google Play, i.e., the app store for android, for many years to disseminate advanced malware [12]. In a recent situation, the android malware 'FlyTrap' app has been utilized to hack numerous Facebook accounts [13]. Additionally, the "Vultur" app was found to be using screen recording features to steal sensitive information. [14]

Various countermeasures have been utilized in literature to mitigate and prevent such malicious attacks. The authors of [15] suggested that there are several kinds of methods to detect malware, including Static analysis, Dynamic analysis, Application permission analysis and Anomaly detection. Each of which has its own technique in order to detect malicious malware, where static analysis depends on extracted features from codes of non-executed apps [16], while, the dynamic analysis relies on monitoring and analyzing the executed apps in a controlled environment. As for application permission analysis, it can be described as a technique considering the access granted from users for android malware. Finally, anomaly detection usually applied using Machine Learning (ML) to identify and predicate malware apps from the learning process and other domains such as, spam detection [17–20], fake news [21]. image segmentation and other fields.

ML malware detection-based acquired more attention from researchers in the past few years due to being more superior compared to other methods. The upgraded malware characteristics made the ML detection-based achieve better performance, because of the periodically updated datasets used in the process. The work in [22], stated that the non-machine learning-based approaches for Android malware detection are consuming more time as well as have less ability to detect malware compared with the ML-based. Also, according to [23], ML detection-based approaches can adapt to the new sophisticated and unpredictable malware more than the other methods. Besides, of all the existing approaches, the ML obtain high accuracy in malware detection [24].

Many works in Android malware detection based on ML have been proposed in the literature, for example, the work in [9] investigated different ML algorithms to detect Android malware. They compared several ML classifiers in the process, including Support Vector Machine (SVM), Naive Bayes (NB), and Random Forest (RF). Another recent work proposed an Android malware detection approach using ML algorithms [25]. The authors apply three different dataset types namely, time-series, Boolean, and frequency datasets. [26] presented four tree-based ML approach for Android malware detection. They used the DREBIN dataset to investigate the performance of the algorithms. The RF achieved the best results when compared with the other methods. Other numerous works also examined the detection of malware utilizing ML algorithms such as [27–29].

Furthermore, the SVM classification model shows outstanding performance against other ML algorithms for Android malicious detection. [30] presented a combination of Active Learning and SVM to detect Android malware. Their approach is evaluated on the DREBIN dataset and shows excellent results in detecting new malicious. Additionally, [31] introduced a keywords correlation distance and SVM method in order to detect Android malware. The method obtained efficient results in detecting malware on Android OS. Another approach tried to detect Android malware using the SVM [32]. In this work, the SVM is grouped with a decision tree (DT) to deal with malware apps. The results demonstrate the superiority of the approach when compared with other detection approaches. Another recent work applied the SVM to detect Android malware using Application Program Interface (API) as features [33]. The results show competitive performance when compared with other approaches.

The SVM shows excellent performance as shown in the previously mentioned works. However, SVM has more space to improve if the optimal hyperparameters were selected [34, 35]. Therefore, in this study, we propose an SVM combined with Harris Hawks Optimization (HHO) for Android malware detection. The HHO in this work, used for two different parts, first, for automatically identifying the best hyperparameters of the SVM, while the second part is for feature weighting to determine the most important ones in order to improve the detection phase. In the proposed approach, the evaluation criteria occurs against the CICMalAnal2017 datasets. Five different datasets are generated from the original data, each data consist of various malware types. Furthermore, additional analysis was applied to describe the relationship and correction of the malware types with the most important features.

In other words, the contribution can be summarized by the following points:

- Android malware detection based on evolutionary Support vector Machine (SVM) algorithm is presented.
- The HHO algorithm is employed to achieve two different objectives simultaneously, including, parameter tuning and feature weighting.
- Generate five various datasets, where each data contain a different malware type.
- Recognizing the most relevant features in order to improve the malware detection phase.

The rest of the paper is organized as follows: Section 2 presents previous works in the literature for Android malware detection. Section 3 introduces the background knowledge of Support Vector Machine (SVM) and Harris Hawks Optimization (HHO). The proposed approach is described in Section 4. Section 5 presents dataset description preparation process. Experiments and results are conducted and described in Section 6, while conclusion and future work is addressed in Section 7.

2 Related work

Android becomes the most used OS in the world in the past few years [3]. The huge number of apps that exist in the Android store reached 2.7 million apps according to [4]. However, not a small number of apps are malware that needs to be detected and controlled. One of the ways to do this is by using malware detection. Android malware detection is the method to distinguish between malicious or benign apps by using techniques such as Static analysis, Dynamic analysis, Application permission analysis, and Anomaly Detection. Machine learning Android malware detection-based gain more attention in recent years.

For instance, the authors of [36] presented a machine learning approach to detect Android malware. They proposed several machine learning methods in order to classify and identify unknown malicious apps. [37] investigated the malware detection using the Significant Permission IDentification (SigPID) system. The SigPID works as a permission technique to analyze and control the increase of Android malware numbers. Further, the system used machine learning methods to classify malware families. Their approach using SVM achieved excellent results in accuracy, recall, precision, and f-measure with a 90% rate with fewer analysis times. On the other hand, the SigPID obtained 93.62% malware detection. Another recent work also applied the machine learning algorithm for Android malware detection [25, 38]. The authors obtain the API information by generating flow graph control of the application separated into three datasets type, time-series, frequency, and Boolean. Based on these datasets, three detection models are constructed for API sequence, API calls, and API frequency. Their ensemble approach examined 10010 benign and 10683 malware and shows excellent results when compared with other methods [39].

Moreover, [40] proposed a machine learning method for Android malware detection based on extracted features of apps. The features are utilized as a set of inputs for the classifier learning. They tried enhancing the detection phase by using the ensemble learning approach (SecENS). Further, the authors develop an efficient system in order to integrate their two methods SecCLS and SecENS together to improve machine learning detection. The work in [41] applied the dynamic analysis for malware detection using machine learning technique. An implementation of automatically feature extracted tool from Android phones was also applied. Their analysis shows that number of features extracted better on-device when compared to emulators and performs better with the machine learning model.

The increasing of Android malware encourages researchers to implement various detection systems. The work in [42] for instance, presented a detection approach of two parts. Firstly, they extract 123 different permissions from more than 10000 applications. Secondly, an evaluation of several machine learning algorithms are applied, namely, Decision Tree (J48), Simple Logistic (SL), k-star, Naive Bayes (NB), and Random Forest (RF). The experiments show that the SL obtained the best results compared with other algorithms. The authors of [43] introduced a lightweight system based on machine learning for Android malware detection. Further, in the system, they used both dynamic and static features alongside the principal component feature selection technique to identify the best set of features. Then the SVM is employed as a classification model. The proposed approach outperforms the other methods [44].

Wang et al. [45] proposed a novel approach for Android malware detection based on information fusion and machine learning methods. The proposed approach applied parallel criteria for the machine learning technique. They start their approach by extracting eight kinds of features, then employed a parallel machine learning model for the purpose of Android malware detection. Additionally, they examine the probability analysis as well as Dempster-Shafer theory on their approach. Another recent work also investigated the Android malware and benign identification using feature weight-based detection, multiple dimensional and kernel feature-based framework [8]. An analysis of 112 data structure kernels in Android OS and examined the detection performance against several types of datasets.

Furthermore, they stated that the memory- and signal-related features obtained the best detection accuracy compared to schedule-related features.

Standard machine learning proved its efficient performance as shown in the previous studies, however, the detection criteria can be improved more using metaheuristic algorithms. Therefore, more recent studies investigated the detection of Android malware combined with metaheuristic algorithms are proposed. For example, [46] proposed a hybrid approach of support vector machine combined with evolutionary algorithms for Android malware detection. This hybrid approach utilizes a genetic algorithm (GA) and a particle swarm optimization (PSO) in order to enhance the detection phase of the SVM. Their approach outperforms the other standard machine learning classifiers. Another recent work applied the metaheuristic and machine learning together for Android ransomware detection [47]. They combined the Kernel Extreme Learning Machine (KELM) with the Salp Swarm Algorithm (SSA) to improve the KELM hyperparameters and select the best subset of features. The experiments of the proposed method achieved better results than other methods in several measures. Furthermore, the work in [48] presented a malware detection technique based on an evolutionary algorithm and operational codes (OpCodes). Their work contained various steps to perform which are, take apart the executable files, producing OpCodes graph, and employing the evolutionary algorithm in order to identify similar graphs. Besides, the detection of the malware types takes place by applying the graph similarity of each instance using the evolutionary algorithm.

Furthermore, a evolutionary algorithm has gain attention recently, Harris Hawks Optimization (HHO). The HHO applied in different and wide applications in the literature, including, feature selection [49], student performance, fault Detection, Internet of Things, image segmentation, manufacturing problems and so on.

Therefore, in this study we utilized the HHO for the problem of malware Android detection. The proposed work in this study differs from the previously mentioned methods in the following points:

- Applied the recent metaheuristic algorithm Harris Hawks Optimization (HHO) in order to improve the SVM detection performance.
- The propped approach HHO-SVM tackles the problem of optimizing the SVM hyperparameters and identify the features weighting of the datasets.
- Generate five sampled datasets to study each scenario of every attack type.
- Analyze each attack type of the five datasets and their relation to the features. In other words, identify the most important features of each dataset (different malware type).

3 Preliminaries

3.1 Support vector machine (SVM)

Support Vector Machine (SVM) is a machine learning classifier designed to solve classification and regression problems [50]. It is known as one of the most reliable classifiers applied for solving problems in different domains [51]. SVM depends on searching for the most possible optimal linear separation criterion which is known as the Hyperplane. The hyperplane tries to maximize the distance (margin) between the closes data points of the training instances which belong to each class. The data points near to the hyperplane in a distance equal to the margin are called Support Vectors. On the other hand, overfitting to the training set will most probably occur, which can lead to the misclassification of the new instances or datasets. To solve this problem, a penalty parameter known as the cost, which is denoted by C, will be used to increase the accuracy of the classification for the new data points. [52]. Figure 1 shows the aforementioned description of SVM.

To solve the issue of nonlinear separation of data points, different kernel functions can be used for SVM. The most popular and robust kernel function is the Radial Basis Function (RBF); as it relies on the gamma γ parameter for

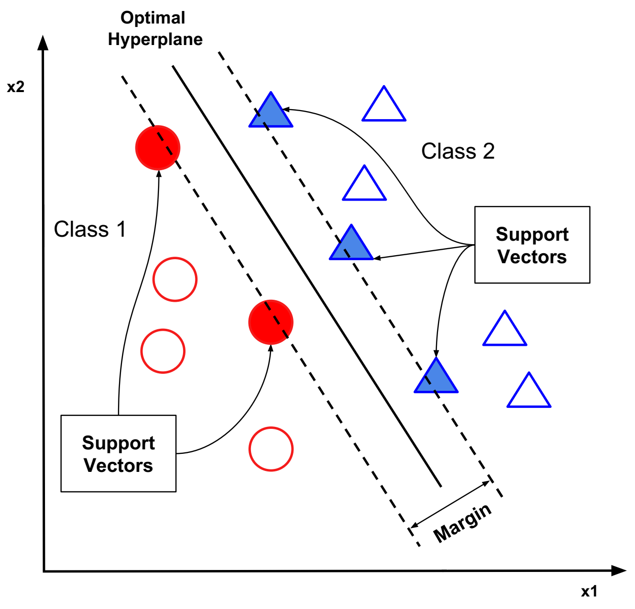


Fig. 1 Support vectors with optimal hyperplane

deciding the effect of support vectors over each other. The reader can refer to [53] for more details about SVM.

3.2 Harris Hawks optimization (HHO)

Harris Hawks Optimizer (HHO) is a nature-inspired population-based optimization algorithm designed by [54]. HHO is inspired by the behavior of Harris' hawks when they cooperate to chase their preys in an intelligent strategy that is called the surprise pounce, through which, the hawks pounce their prey from different directions to surprise it as shown in Fig. 2.

As a nature-inspired optimization algorithm, the HHO is composed of two main phases; which are the exploration and exploitation, as well as a transition state between exploitative behaviors. The candidate solutions here are represented by the hawks that are observing and waiting in the desert to detect their prey, and the best solution for each step is the selected prey.

In the exploration phase, the Harris' hawks start their haunting process by selecting random locations and waiting to try to detect a prey. This is carried out based on two strategies: the first depends on the positions of other hawks that are participating in the haunting of the prey, and the second depends on the randomly existing tall trees within the haunt range. Equation 1 explains both strategies, where an equal chance q for every positioning (perching) strategy is considered and thus, the first strategy is selected if q is equal to or greater than 0.5, and the second strategy is selected otherwise. $X(t+1)$ is the vector of hawks' positions in the following iteration, $X_{rabbit}(t)$ is the position of the prey in the current iteration t, $X_{rand}(t)$ is a randomly selected hawk from the current iteration, and $X(t)$ is the vector of hawks' positions of the current iteration. r_1, r_2, r_3, r_4 and q are random numbers in the interval (0,1) which are updated through each iteration, LB and UB are the lower and upper bounds of the variables, respectively.

$$x(t+1) = \begin{cases} X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)| & q \geq 0.5 \\ (X_{rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases} \tag{1}$$

$X_m(t)$ is the average position of hawks in the current population, which can be calculated according to equation 2, where $X_i(t)$ is the position of the hawk i in the current iteration, while N is the total number of hawks.

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \tag{2}$$

In the exploitation phase, the Harris' hawks start attacking their prey by performing the surprise pounce. However, as the prey attempts several times to escape from the hawks, they change their chasing strategies according to the escaping behaviors of the prey. Hence, there are four different chasing strategies followed by the hawks, which are Soft Besiege, Soft Besiege with progressive rapid dives, Hard Besiege, Hard Besiege, and Hard Besiege with progressive rapid dives.

The selection of either strategy of the four depends on the energy E of the prey; as the prey loses its energy during escaping the haunt. In other words, it can be translated as changing between different exploitative behaviors. The energy of the prey can be modeled through Equ 3, where E_0 is the initial energy of the prey, and T is the maximum number of iterations.

$$E = 2E_0\left(1 - \frac{t}{T}\right) \tag{3}$$

When $|E| \geq 0.5$, the soft besiege occurs only if the chance r of the prey successfully escaping from the hawks is ≥ 0.5. However, if $r < 0.5$, the soft besiege with progressive rapid dives strategy occurs. Equ 4 and 5 illustrate both strategies, respectively. Where $\Delta X(t)$ is the difference between the position vector of the rabbit and the location stored in the current iteration t, Y is the rule to evaluate the next move of the hawks to perform a soft besiege. While Z is the rule to apply the zigzag deceptive motion that is mimicked in the Levy Flight LF move, only if the Y rule fails. The reader can read about Y, Z, and LF in original paper.

$$X(t+1) = \Delta X(t) - E|JX_{rabbit}(t) - X(t)| \tag{4}$$

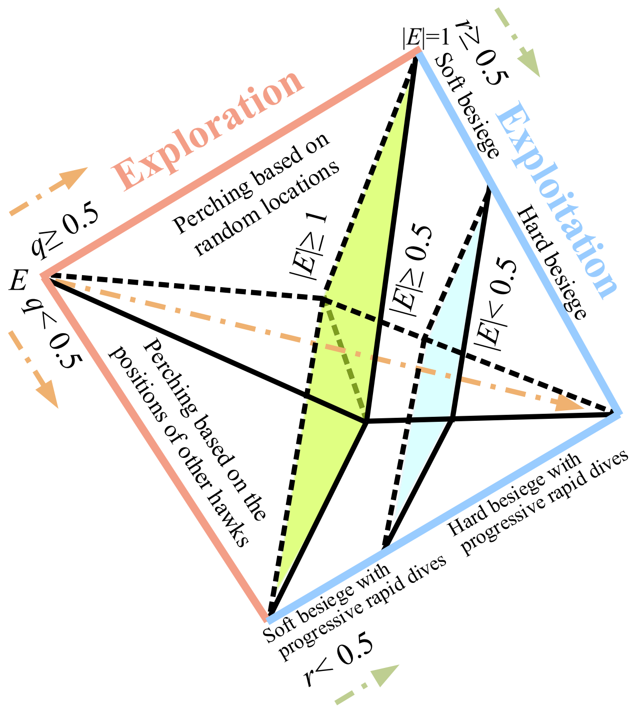

|E|=1
r≥ 0.5
Soft besiege
Exploitation
Hard besiege
Exploration
Perching based on random locations
q≥ 0.5
E
q< 0.5
Perching based on the positions of other hawks
|E|≥ 1
|E|≥ 0.5
|E|< 0.5
Soft besiege with progressive rapid dives
Hard besiege with progressive rapid dives
r< 0.5

Fig. 2 HHO different phases [54]

$$X(t+1) = \begin{cases} Y & \text{if } F(Y) < F(X(t)) \\ Z & \text{if } F(Z) < F(X(t)) \end{cases} \quad (5)$$

When $|E| < 0.5$, the Hard besiege strategy is followed in condition that r is greater than or equal to 0.5. Otherwise, hard besiege with progressive rapid dives strategy will be carried out. Equation 6 shows how the current positions are updated for hard besiege, while for hard besiege with progressive rapid dives the same Equ 5 is applied with a difference that Y considers the average positions of the hawks instead.

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \quad (6)$$

4 Proposed approach

4.1 Design issues

This section detailedly describes, the approach followed to utilize the HHO algorithm for optimizing SVM parameters as well as weighting the features.

Solution representation As described earlier, in this paper, the solution here is represented by the hawks that are waiting and observing the prey. Therefore, from now on, we will only mention the term solution instead of hawks to eliminate any confusion for the reader. The representation of the solution is affected by two factors; the parameters of SVM, and the features (attributes) of the inserted dataset.

For the first part of the representation, we look into the search spaces of SVM parameters C and γ, which both have different boundaries than the original boundaries of the solution, which are 0 and 1. Therefore, we need to scale the values of the solution into readable values for SVM parameters. The C parameter best accepts values between 0 and 32, and γ values can be within the interval [0,35000]. For scaling the values, we apply the min-max normalization equ shown in 7, where B represents the final scaled value, A is the value to be scaled, min_A, max_A are the lower and upper bounds of the old interval, respectively, and min_B, and max_B are the lower and upper bounds of the new interval, respectively.

$$B = \frac{A - min_A}{max_A - min_A}(max_B - min_B) + min_B \quad (7)$$

The second part of the solution representation is directly affected by the number of attributes of the dataset. Each value in the solution vector is matched to an attribute in the dataset. Consequently, the value of each cell in the second part of the solution that is produced by the HHO algorithm is multiplied by the value of the corresponding attribute for all training instances as shown in Fig. 3. Hence, combining both parts will result in a solution that has a length that is equal to two (for C γ) plus the number of the attributes of the dataset.

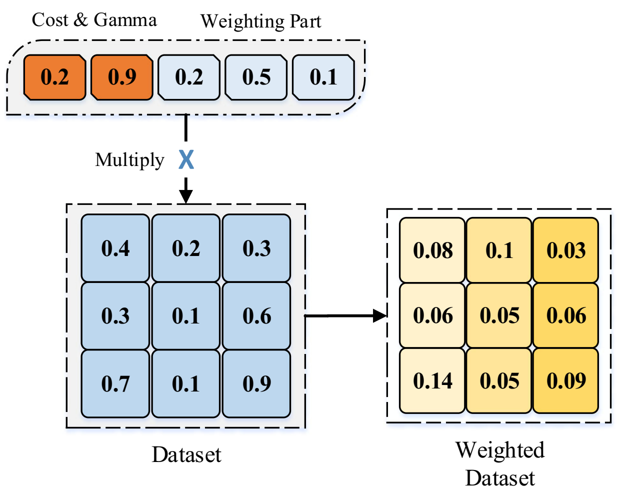


Fig. 3 Representation of the weighting mechanism

Fitness function Deciding the quality of the solution is known as fitness assessment and thus, the function that is used for such task is called the fitness function. In our case, we use the classification accuracy produced by the SVM algorithm as the fitness function, and we make sure that HHO is also set to maximize this value. The classification accuracy is calculated using Eq. 8, where TP and TN are the truly classified positive and negative instances, and FP and FN are the falsely classified positive and negative instances.

System architecture To start the process, we split the dataset into training and testing subsets. The split is conducted based on the k-fold cut, where the $k-(k-1)$ partition is allocated for the testing set and the remaining $k-1$ is allocated for the testing set. This step is repeated k times, having different $k-1$ parts of the dataset for both training and testing sets in each iteration. This step is conducted to guarantee the maximum possible diversity of the training/testing sets, as well as maximum possible number of separated runs.

The HHO initializes a random solution based on the training set at the beginning of each fold. The solution will be composed of the values that will be given to the SVM parameters and the features of the training dataset. The solution is split by assigning the first two values for SVM parameters after having them scaled and the remaining part will be assigned to the features. The value in each cell in the second part of the solution is multiplied by each value of the matching feature.

Next, the SVM is trained using the scaled values of the first two cells of the solution vector which are assigned to the C and γ variables, as well as the new values of the training set which resulted from the multiplications by the correspondent cells. The classification accuracy that resulted using the values of the solution is returned as the fitness function outcome for the HHO algorithm.

As we mentioned, all previous operations occur during a single training fold and they are repeated in that fold based on the number of iterations set in the HHO algorithm. When the maximum number of iterations is reached, the HHO returns the most optimal possible solution which owns the highest classification accuracy, and this value will be the outcome of that single fold. Finally, we calculate the average accuracy out of the accuracy of the testing set of all folds.

4.2 Evaluation

The results of all algorithms are compared and evaluated in order to examine their performance on the datasets. The evaluation process performed using the confusion matrix table as illustrated in Fig. 4. The True Positive (TP) depicts the number of all actual positive classes that are accurately predicted, while the count of actual positive elements that are incorrectly predicted denotes with False Negative (FN). As for False Positive (FP), it is the number of negative elements that are incorrectly predicted as Positive class, whilst the count of negative elements that are accurately predicted.

Four evaluation measures are utilized to examine the model's performance, including, accuracy, precision, recall, and f-measure. The mentioned measures can be calculated as shown in the following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall(Sensitivity) = \frac{TP}{TP + FN} \tag{10}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{11}$$

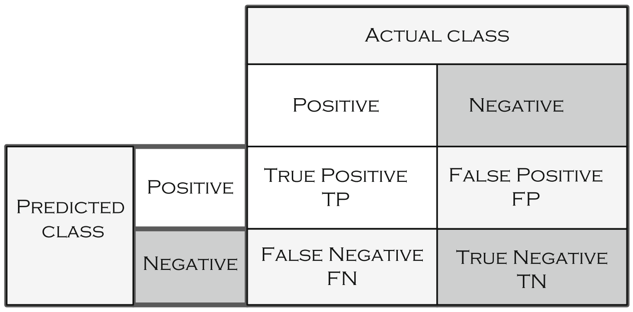


Fig. 4 Confusion matrix

All the aforementioned processes described in Sect. 4 are depicted in Fig. 5.

5 Dataset description, characteristics and preparation

The dataset used in this work (CICAndMal2017) is collected from 10,854 samples by the Canadian Institute for Cybersecurity [55]. The samples consist of 4354 malware and 6500 benign gathered from different sources. On one hand, the malware sources are divided into several parts, namely, Contagio security blog, VirusTotal, and previously published works in the literature [56], and on the other hand, the benign application samples are collected from Google Play store during the years 2015-2017.

The dataset is categorized into five different groups, including, Benign, Adware, Ransomware, Scareware and SMSmalware. The details of each group can be seen in Table 1.

The Adware type is defined as a malicious application that is responsible for sending user information to a specific remote server in order to forcefully showing personalized (interest-based) advertisements for that user. This can be done by hacking smartphone speakers or tracking users' search history and application usage [57]. As for the Ransomware type, it is a kind of malware that demands users an amount of cash. This type has two general classes, which are, crypto and lock-screen. The crypto class works as an encryption method that scrambles the mobile device information and contents, while the lock-screen class functions by blocking the smartphone screen and covers it completely with a picture and make it impossible to be used. Both classes can be resolved if the users pay the demanded payment. Moreover, the Scareware type operates by scaring users with some kind of phishing websites or applications that threaten them to steal their information [58]. Scareware tries to trick users by pretending to be a security application, for instance, that shows a fake list of viruses on users' devices causing them to use such malware. Finally, the SMSmalware type is a malware that controls and manages messages to send unwanted messages. In other words, it is a process used by the hacker to send messages by using users' mobile phones to trick their trusted contacts [59].

Before processing, the dataset comprises various files where each file is a member of the malware groups (Adware, Ransomware, Scareware, and SMSmalware). Therefore, to merge all files together, a command line script was applied. Further, some prepossessing steps were employed in order to prepare the dataset for the classification models, including, clean noisy data and missing

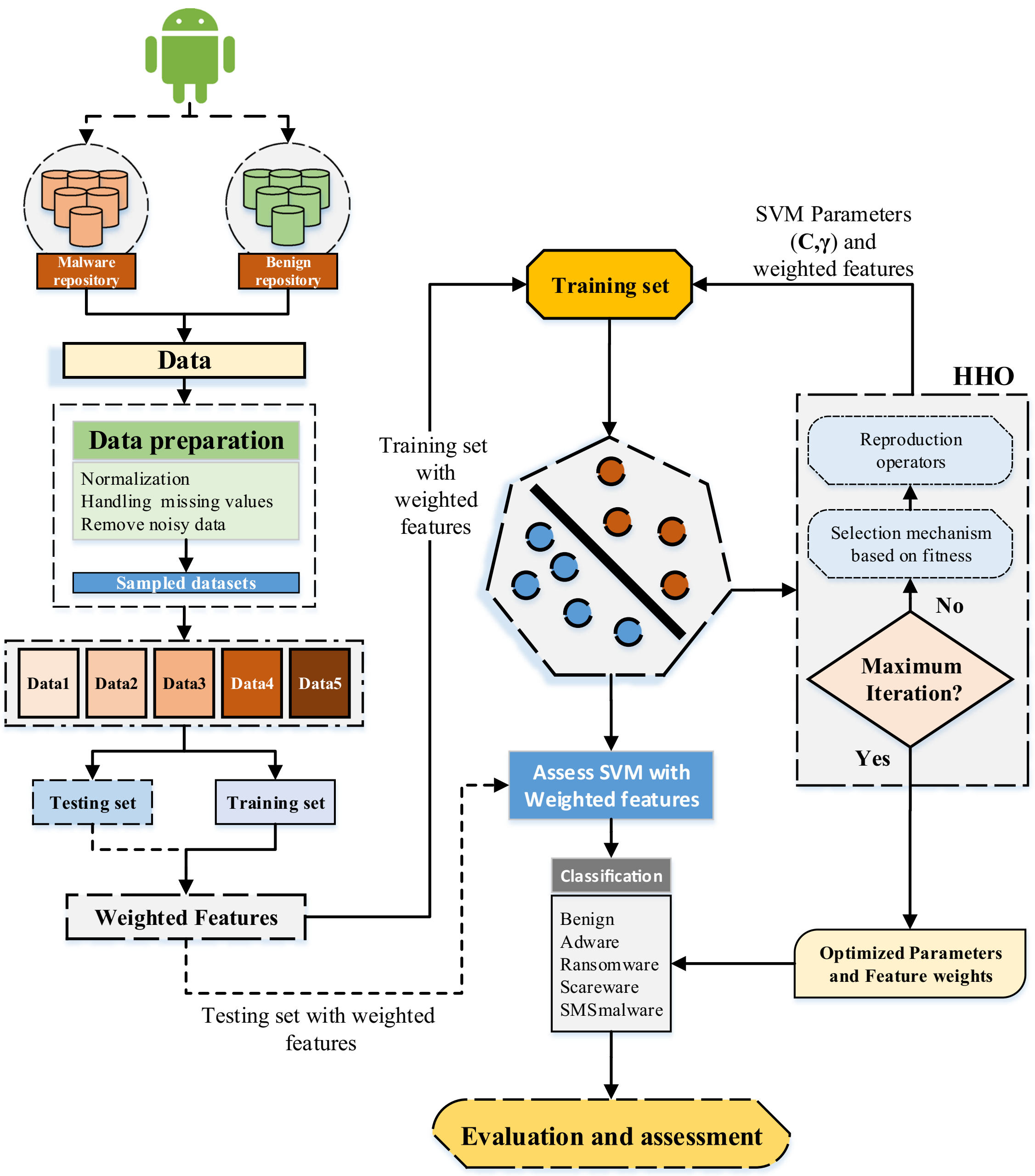


Fig. 5 Proposed approach process

Table 1 Details of the original data

Type	No. of instances
Benign	1,210,210
Adware	424,147
Ransomware	348,943
Scareware	401,165
SMSmalware	229,275

values. These issues were solved using normalization and majority vote methods, respectively.

In this paper, five datasets are prepared and sampled from the original data. In our sampling technique, we generated each dataset to have two types of classes, Benign and different malware types, while the last dataset contains all malware types except the Adware due to having diverse characteristics. Also, all datasets simulate the distribution of the original data. Additional experiments were applied with other feature selection approaches and the same results were obtained. The details of the sampled datasets are shown in Table 2.

6 Experiment and results

In this section, several experimental phases are carried out on all datasets, including, base classifiers models examination, SVM with metaheuristic benchmarks investigation, comparison between the proposed approach, and other metaheuristic algorithms on our datasets.

Additionally, feature importance analysis is applied in order to identify the best-weighted ones to detect android malware on each dataset.

Therefore. to summarize the four experiments and analysis phases:

– Base classifiers models: an examination of our sampled datasets on traditional well-known classification models.
– Benchmarks performance of SVM with metaheuristic algorithms: to investigate the performance of HHO-SVM against other algorithms with distinguished benchmarks.
– HHO-SVM against other metaheuristic algorithms: an examination of our sampled datasets on the proposed HHO-SVM compared with other algorithms.
– Feature Importance analysis: an analysis of the most important features to detect malware as well as the relationship between each class type and the features.

6.1 Experiments setup

All experiments were conducted on a workstation with the specification of Xeon E5-2609 CPU and 64GB RAM. All algorithms were implemented on MATLAB 2016 version A. As for the base classifiers we used Weka tool. The settings and parameters of the metaheuristic algorithms can be found in Table 3.

Furthermore, 10 independent runs are conducted for all approaches and the average alongside standard deviations of the runs were taken, while the number of iteration was 20 for metaheuristic algorithms. The performed measures in this work are accuracy, precision, recall, and f-measure. As for the training and testing splitting criteria, we utilized the 10-fold cross-validation, thus, we guarantee the maximum shuffle for the testing and training sets. Finally, all approaches are examined on the 5 sampled datasets, and an extra investigation of the metaheuristic is examined on 10 well-known benchmarks.

Table 3 Parameter settings

Algorithm	Parameter	Value
SSA	c_1	[0-1]
	c_2	[0-1]
PSO	Acceleration constants	[2.1, 2.1]
	Inertia w	[0.9, 0.6]
GA	Single point crossover	0.9
	Mutation	0.01

6.2 Performance of the sampled data (CICMalAnal2017) on the base classifiers models

In the first phase, the performance of the base classifiers models is investigated on the new 5 sampled datasets. Theses classifiers are commonly used in the literature, which are, Naive Bayes (NB), k-nearest Neighbors (k-NN) with different k values and Random Forest (RF). This phase implemented in order to analyze and examined the sampled datasets before executed on our proposed approach.

Table 4 illustrates the results for $Data_1$ dataset. The highest results obtained by 5-NN with 85.57%, while the second highest achieved by NB classifier with 84.86%. In terms of precision, the NB achieved the highest result with 0.876%, followed by RF with 0.8745%. As for the recall measure, the maximum result accomplished by 5-NN with 0.97%, while the second best acquired by NB. If we observe the f-measure results, the 5-NN has the best result, followed by NB, RF, 3-NN and 1-NN, respectively.

Table 2 Details of the five sampled datasets

Dataset	No. of instances	Name of classes
$Data_1$	2400	Benign, Adware
$Data_2$	1725	Benign, Ransomware
$Data_3$	2210	Benign, Scareware
$Data_4$	2450	Benign, SMSmalware
$Data_5$	2850	Benign, Ransomware, Scareware, SMSmalware

Table 4 Base classifiers results for $Data_1$ dataset

Dataset	Algorithm	Accuracy		Precision		Recall		F-measure	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std
$Data_1$	NB	84.8689	1.6960	0.8767	0.0058	0.9620	0.0190	0.9173	0.0100
$Data_1$	1-NN	77.4990	2.0221	0.8723	0.0082	0.8697	0.0222	0.8708	0.0129
$Data_1$	3-NN	84.0225	1.3189	0.8741	0.0054	0.9545	0.0136	0.9125	0.0077
$Data_1$	5-NN	**85.5772**	1.1717	0.8721	0.0036	0.9783	0.0122	0.9221	0.0068
$Data_1$	RF	84.5644	1.3087	0.8745	0.0049	0.9611	0.0138	0.9158	0.0076

The bold values represent the highest results

Based on the results for $Data_2$ in Table 5, we notice that the NB outperforms all other methods with 86.7% in terms of accuracy, followed by 5-NN, RF, 3-NN and 1-NN, respectively. As for the precision results, the 1-NN provides the fittest result, while 3-NN obtained the second best result. In terms of recall and f-measure, the best classifier was NB with 0.99% and 0.92%, respectively.

According to the accuracy results for $Data_3$ in Table 6, 5-NN achieved the best results with 92.6%, followed by 3-NN, RF, 1-NN and NB, respectively. As for the precision measure, the NB outperforms other classifiers, while the highest result in recall and f-measure accomplished by 5-NN with 0.99% and 0.96%, respectively.

The 5-NN achieved the highest accuracy for $Data_4$ as shown in Table 7, followed by 3-NN, RF, NB and 1-NN, respectively. As for the precision measure, the 3-NN exceeds all other classifiers, while 5-NN is placed second with 0.9313%. In terms of recall and f-measure, the 5-NN obtained the highest results with 0.99% and 0.96%, respectively.

As per accuracy results in Table 8, 5-NN outperforms all other methods, followed by RF, 3-NN, NB and 1-NN, respectively. NB classifier provides the fittest results compared to the other classifiers in terms of precision with 0.878%. As for recall and f-measure, 5-NN also acquired the highest results, while the second best achieved by RF.

In summary, the investigation of the datasets shows the stability of the results in total. Therefore, the examination of our proposed approach can be employed after this analysis. It is worth mentioned that the best algorithm obtained by the 5-NN, where it acquired first place 4 times in terms of accuracy, while NB achieved the first place in one time. As for the other measure, NB placed first 3 times in precision and 5-NN placed first for recall and f-measure in 4 times.

6.3 Performance of HHO-SVM on general benchmarks

Before examining our sampled datasets on the the proposed approach, a general performance investigation is applied on number of benchmarks. This is done in order to verify the proposed approach performance compared with the other algorithms. Table 9 reports a brief description of the 10 utilized benchmarks in this subsection.

Four metaheuristic algorithms are applied in this phase which are, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Salp Swarm Algorithm (SSA), and Harris Hawk Optimizer (HHO).

According to the average accuracy shown in Table 10, HHO-SVM outperforms the other algorithms with 6 out of 10 datasets, namely, Breast Cancer, Wine, Sonar, Spectft, Ionosphere, Glass and Iris. Further, Parkinsons dataset achieved the best result by both the HHO-SVM and GA-SVM with 94.8421%. The two other datasets, Heart and

Table 5 Base classifiers results for $Data_2$ dataset

Dataset	Algorithm	Accuracy		Precision		Recall		F-measure	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std
$Data_2$	NB	**86.7745**	0.7019	0.8724	0.0018	0.9937	0.0084	0.9291	0.0041
$Data_2$	1-NN	78.9731	2.3607	0.8795	0.0088	0.8795	0.0265	0.8793	0.0150
$Data_2$	3-NN	83.7240	1.8795	0.8763	0.0077	0.9471	0.0186	0.9103	0.0109
$Data_2$	5-NN	85.1909	1.2185	0.8720	0.0037	0.9731	0.0134	0.9197	0.0071
$Data_2$	RF	84.7673	1.4224	0.8731	0.0049	0.9658	0.0153	0.9170	0.0083

The bold values represent the highest results

Table 6 Base classifiers results for $Data_3$ dataset

Dataset	Algorithm	Accuracy		Precision		Recall		F-measure	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std
$Data_3$	NB	87.0367	2.1158	0.9314	0.0055	0.9287	0.0226	0.9299	0.0122
$Data_3$	1-NN	87.2857	1.5966	0.9304	0.0052	0.9328	0.0159	0.9315	0.0091
$Data_3$	3-NN	91.7837	0.7733	0.9291	0.0036	0.9868	0.0075	0.9570	0.0042
$Data_3$	5-NN	**92.6245**	0.3859	0.9285	0.0025	0.9974	0.0037	0.9617	0.0021
$Data_3$	RF	91.5592	0.7845	0.9283	0.0030	0.9851	0.0077	0.9558	0.0043

The bold values represent the highest results

Table 7 Base classifiers results for $Data_4$ dataset

Dataset	Algorithm	Accuracy		Precision		Recall		F-measure	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std
$Data_4$	NB	88.7376	2.2982	0.9304	0.0040	0.9501	0.0244	0.9400	0.0129
$Data_4$	1-NN	87.3394	1.6832	0.9310	0.0042	0.9332	0.0184	0.9320	0.0097
$Data_4$	3-NN	92.1946	0.6507	0.9316	0.0028	0.9888	0.0068	0.9593	0.0035
$Data_4$	5-NN	**92.9050**	0.3958	0.9313	0.0021	0.9973	0.0037	0.9632	0.0021
$Data_4$	RF	91.7602	0.7481	0.9303	0.0018	0.9854	0.0082	0.9570	0.0041

The bold values represent the highest results

Table 8 Base classifiers results for $Data_5$ dataset

Dataset	Algorithm	Accuracy		Precision		Recall		F-measure	
		Avg	Std	Avg	Std	Avg	Std	Avg	Std
$Data_5$	NB	84.1455	2.0462	0.8784	0.0048	0.9509	0.0218	0.9131	0.0121
$Data_5$	1-NN	78.4169	1.8981	0.8763	0.0069	0.8779	0.0207	0.8770	0.0119
$Data_5$	3-NN	84.4436	1.2187	0.8765	0.0038	0.9576	0.0130	0.9152	0.0071
$Data_5$	5-NN	**86.9146**	0.6112	0.8780	0.0024	0.9881	0.0064	0.9298	0.0034
$Data_5$	RF	85.2896	1.0068	0.8776	0.0036	0.9671	0.0109	0.9202	0.0058

The bold values represent the highest results

Table 9 List of benchmark datasets

No.	Dataset	No. of features	No. of instances	No. of classes
1.	Breast Cancer	10	683	2
2.	Wine	13	178	3
3.	Heart	13	270	2
4.	Parkinsons	22	195	2
5.	Sonar	60	208	2
6.	Vowel	10	528	11
7.	Spectft	44	276	2
8.	Ionosphere	34	351	2
9.	Glass	9	214	6
10.	Iris	4	150	3

Table 10 A comparison of average accuracy for GA-SVM, PSO-SVM, SSA-SVM and HHO-SVM over all benchmarks

Algorithm	GA-SVM		PSO-SVM		SSA-SVM		HHO-SVM	
Benchmark	Avg	Std	Avg	Std	Avg	Std	Avg	Std
Breast Cancer	96.3342	1.9904	96.7796	2.1518	96.6326	2.7638	**96.9224**	1.6174
Wine	97.2222	2.9280	97.7778	3.8845	97.1895	2.9641	**98.3007**	2.7377
Heart	80.3704	8.0104	80.7407	11.4223	**82.9630**	7.2409	82.5926	9.0806
Parkinsons	**94.8421**	6.7575	92.2368	8.2970	93.4211	5.7762	**94.8421**	5.9289
Sonar	83.1905	7.4849	87.9762	6.8413	86.5476	11.2168	**88.0238**	5.5579
Vowel	98.8643	2.0298	**99.4340**	0.9114	98.8643	1.3216	99.0530	1.3371
Spectft	78.2764	5.8175	78.6610	7.8639	78.9744	9.8853	**79.3875**	9.2618
Ionosphere	92.8810	7.4031	92.3175	2.2980	92.0079	4.2272	**93.1508**	4.0940
Glass	65.9524	9.1719	66.8831	8.3825	69.6970	11.9965	**71.9913**	8.4068
Iris	94.6667	2.8109	93.3333	4.4444	94.0000	6.6295	**95.3333**	5.4885
Rank	4th		2nd		3rd		1st	

The bold values represent the highest results

Vowel, obtained the highest results by SSA-SVM and PSO-SVM, respectively.

The previous results clearly show the superiority of the HHO-SVM compering with other approaches on the 10 benchmarks. However, for more accurate examination, all four approaches are also compared with our sampled datasets.

6.4 Results of HHO-SVM compared with other metaheuristic algorithms on the sampled data (CICMalAnal2017)

In this subsection, the HHO-SVM applied on the sampled datasets and compared with the remaining metaheuristic algorithms. These algorithms are the same as the previous subsection, including GA-SVM, PSO-SVM, and SSA-SVM. Moreover, unlike the previous phase, the examination take place on several measures namely, accuracy, , recall, precision, and f-measure.

Table 11 illustrates the results of the four algorithms in terms of all measures for the $Data_1$ dataset. In terms of accuracy, HHO-SVM achieved the best result, followed by GA-SVM, PSO-SVM, and SSA-SVM, respectively. Regarding the recall measure, HHO-SVM obtained the highest result, while GA-SVM placed second. Further, the HHO-SVM also acquired the fittest result for precision and f-measure with 99.95% and 93.20%, respectively, while the best-second achieved by PSO-SVM for both measures.

As shown in Table 12, the HHO-SVM exceeds all other approaches in terms of accuracy for $Data_2$, followed by PSO-SVM, SSA-SVM, and GA-SVM. As per recall results, the best result is obtained by HHO-SVM, while PSO-SVM has the second highest result. According to the precision measure, the best results gained by both HHO-SVM and SSA-SVM. The PSO-SVM has the fittest result in f-measure compared to other algorithms with 93.14%.

Table 13 states the comparison of all algorithms for $Data_3$ dataset. As per results for accuracy, recall and precision measures, we can see that all algorithms have the same results. This is happened due to the sensitivity of the dataset and the distribution of the classes. As for f-measure, the PSO-SVM attained the best result with 96.4355%.

Table 14 reflects the comparison results of $Data_4$ data-set. The HHO-SVM reached the highest accuracy rates with 92.85%, which is followed by PSO-SVM, SSA-SVM and GA-SVM, respectively. Regarding the recall and f-

Table 11 HHO-SVM and other metaheuristic algorithms results for $Data_1$ dataset

Dataset	Algorithm	Accuracy		Recall		Precision		F-measure		Rank
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	
$Data_1$	GA-SVM	87.2448	2.0984	87.3162	2.0243	99.9029	0.30702	93.1758	1.2059	2nd
$Data_1$	PSO-SVM	87.2444	1.9681	87.3161	1.9079	99.9031	0.20423	93.1771	1.1198	3rd
$Data_1$	SSA-SVM	87.2019	2.4614	87.3113	2.4652	99.8572	0.31781	93.1469	1.4017	4th
$Data_1$	HHO-SVM	**87.2868**	1.7891	**87.3220**	1.7231	**99.9507**	0.15578	**93.2031**	1.0208	1st

The bold values represent the highest results

Table 12 HHO-SVM and other metaheuristic algorithms results for *Data*$_2$ dataset

Dataset	Algorithm	Accuracy		Recall		Precision		F-measure		Rank
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	
Data$_2$	GA-SVM	87.1216	2.2218	87.2217	2.1654	99.8658	0.2830	93.1043	1.2512	4th
Data$_2$	PSO-SVM	87.1787	1.5535	87.2287	1.4993	99.9324	0.2137	**93.1437**	0.8831	2nd
Data$_2$	SSA-SVM	87.1757	2.5126	87.2264	2.5055	**99.9333**	0.2108	93.1310	1.4472	3rd
Data$_2$	HHO-SVM	**87.1841**	2.4637	**87.2348**	2.4562	**99.9333**	0.2108	93.1366	1.4116	1st

The bold values represent the highest results

Table 13 HHO-SVM and other metaheuristic algorithms results for *Data*$_3$ dataset

Dataset	Algorithm	Accuracy		Recall		Precision		F-measure		Rank
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	
Data$_3$	GA-SVM	**93.1222**	1.5352	**93.1222**	1.5352	**100.0000**	0.0000	96.4327	0.8273	1st
Data$_3$	PSO-SVM	**93.1222**	1.1247	**93.1222**	1.1247	**100.0000**	0.0000	**96.4355**	0.6025	1st
Data$_3$	SSA-SVM	**93.1222**	1.2401	**93.1222**	1.2401	**100.0000**	0.0000	96.4348	0.6653	1st
Data$_3$	HHO-SVM	**93.1222**	1.1247	**93.1222**	1.1247	**100.0000**	0.0000	96.4354	0.6070	1st

The bold values represent the highest results

Table 14 HHO-SVM and other metaheuristic algorithms results for *Data*$_4$ dataset

Dataset	Algorithm	Accuracy		Recall		Precision		F-measure		Rank
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	
Data$_4$	GA-SVM	92.8163	1.2047	92.8488	1.1723	99.9558	0.1399	96.2680	0.6416	2nd
Data$_4$	PSO-SVM	92.7755	1.4014	92.7755	1.4014	**100.0000**	0.0000	96.2475	0.7532	3rd
Data$_4$	SSA-SVM	92.7347	1.2734	92.7727	1.2764	99.9561	0.1387	96.2263	0.6886	4th
Data$_4$	HHO-SVM	**92.8571**	2.3741	**92.8526**	2.3711	**100.0000**	0.0000	**96.2796**	1.2853	1st

The bold values represent the highest results

measures, the HHO-SVM obtained the best results with 92.85% and 96.27%, respectively. As for the precision, we observe that the HHO-SVM joint with PSO-SVM achieved the highest result.

As per results for *Data*$_5$ dataset in Table 15, it is observed that the HHO-SVM again exceeds the other approaches in terms of accuracy. The GA-SVM, PSO-SVM and SSA-SVM has attained the next rates,

Table 15 HHO-SVM and other metaheuristic algorithms results for *Data*$_5$ dataset

Dataset	Algorithm	Accuracy		Recall		Precision		F-measure		Rank
		Avg	Std	Avg	Std	Avg	Std	Avg	Std	
Data$_5$	GA-SVM	87.6794	1.4886	**87.7111**	1.5499	**99.9612**	0.1226	**93.4293**	0.8409	2nd
Data$_5$	PSO-SVM	87.6102	1.9331	87.7023	1.9128	99.8798	0.2707	93.3858	1.1040	3rd
Data$_5$	SSA-SVM	87.6084	1.8844	87.6998	1.8326	99.8789	0.19505	93.3852	1.0781	4th
Data$_5$	HHO-SVM	**87.6803**	2.1189	87.7103	2.0827	99.9590	0.1296	93.4235	1.2087	1st

The bold values represent the highest results

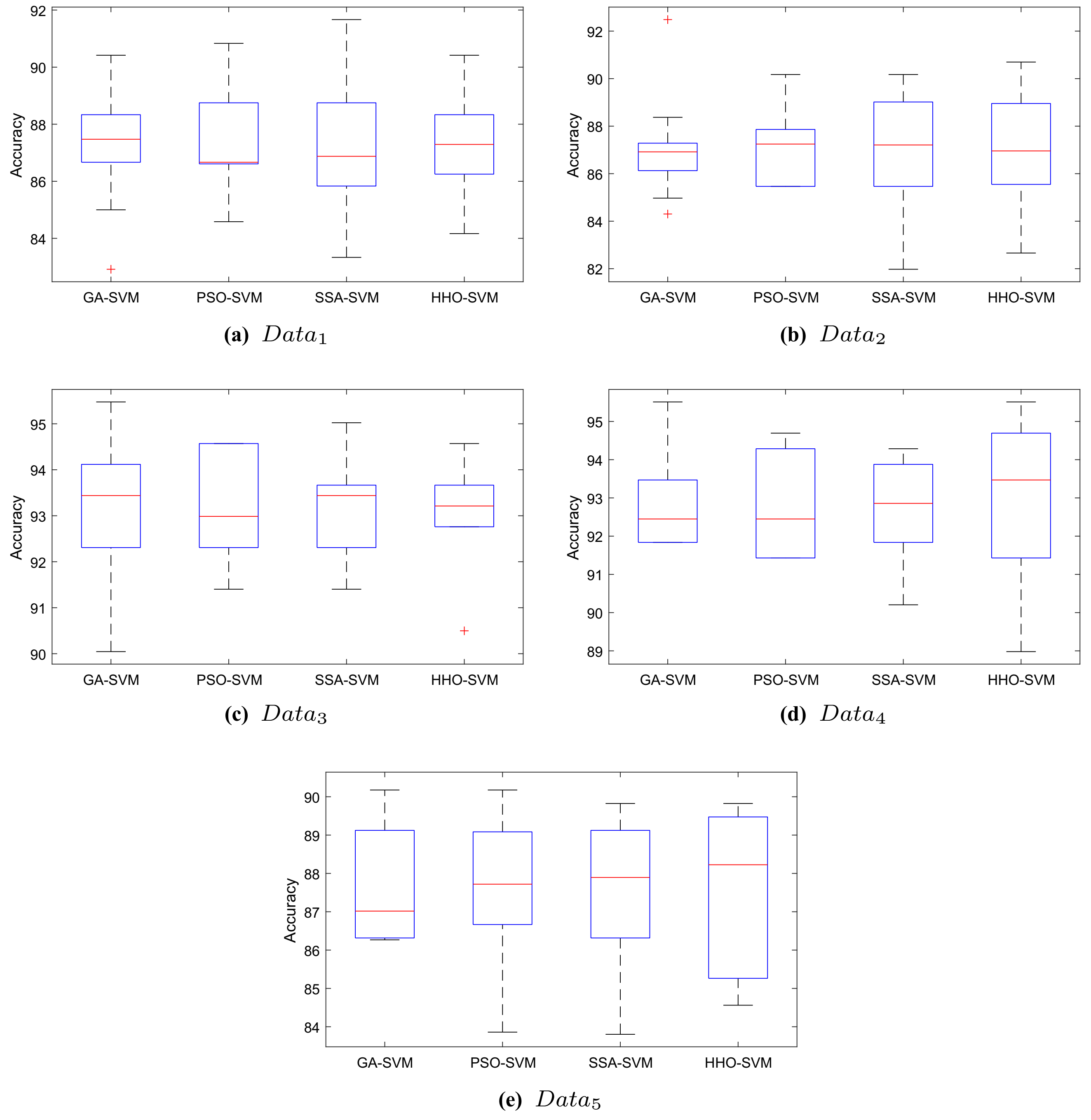
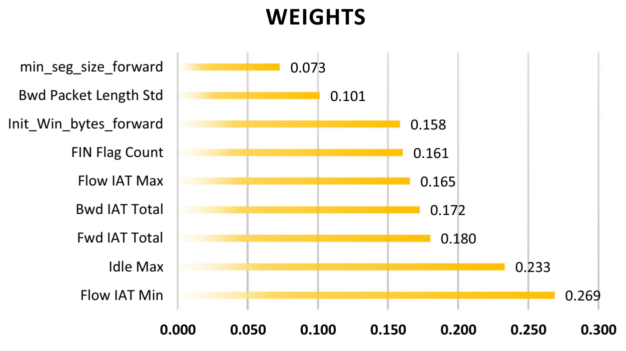


Fig. 6 Box-plot charts for HHO-SVM and other algorithms based on sampled-datasets

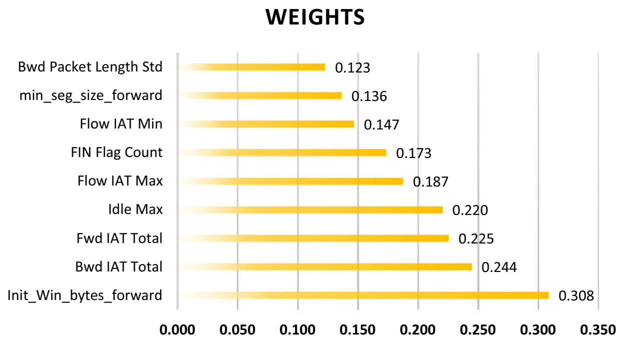
respectively. However, in terms of recall, precision, and f-measure, the GA-SVM attained the best result with 87.711%, 99.96% and 93.429%, respectively. While the HHO-SVM obtained the second best result for the three measures.

Mainly, the results of SVM in total shows improvement compared with the base classifiers phase. All metaheuristic algorithms obtained good results, however, the HHO-SVM outperforms all other approaches in most of the measures. This again proves the superiority of the proposed approach (HHO-SVM).

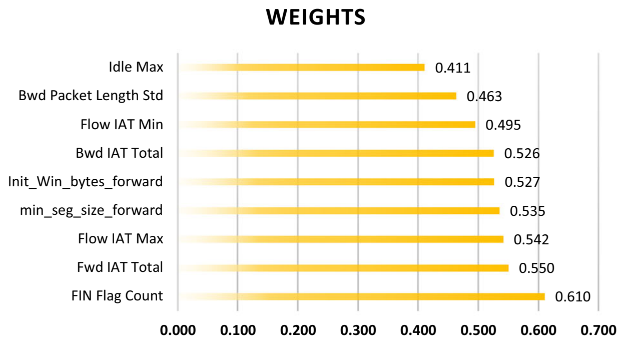
Figure 6 presents the box-plot charts for all datasets in terms of accuracy. The box-plots is determined by using the 10-runs values of the accuracy measure for each algorithm.



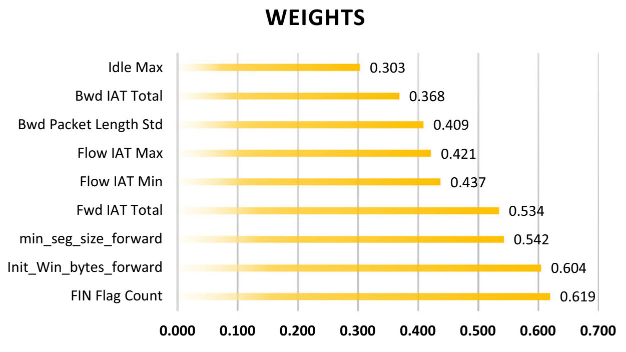
(a) $Data_1$



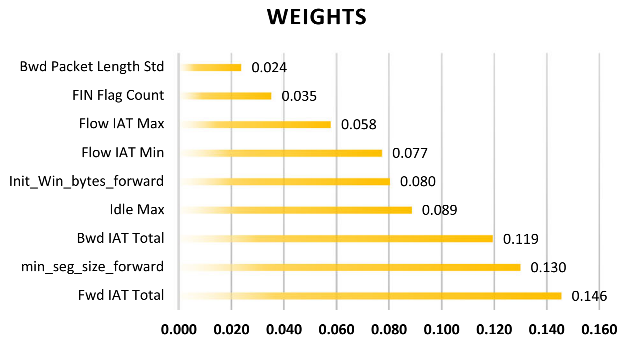
(b) $Data_2$



(c) $Data_3$



(d) $Data_4$



(e) $Data_5$

Fig. 7 HHO-SVM feature weighting for all datasets, incdluding $Data_1$, $Data_2$, $Data_3$, $Data_4$ and $Data_5$

6.5 Feature importance analysis

In this subsection, a feature importance analysis is presented in order to investigate the features' weights for each dataset. This analysis will help us to identify the relationship between the features and the types of classes that each dataset has. Therefore, more identification and explanation is needed about the most important features to detect the malware for each scenario.

According to Fig. 7, the weights of the features reveals different values for each datasets. In Fig. 7a for example, the best feature was $min_seg_size_forward$, while the second best feature was *IdleMax*. In $Data_2$, the highest weighted feature was *Init Win bytes forward* as shown in Fig. 7b, and the second obtained by *Bwd IAT Total*. As can be seen in Fig. 7c the first and second features were *FIN Flag Count* and *Fwd IAT Total* with 0.610 and 0.550, respectively. On the other hand, Fig. 7d illustrates the features' weights for $Data_4$, where *FIN Flag Count* achieved the first place and *Init Win bytes forward* acquired the second place. Finally, Fig. 7e shows the weights of $Data_5$ features. *Fwd IAT Total* attained the highest weighted feature, while *Init Win bytes forward* was the second highest.

Overall, 7 reveals each dataset's important features and the difference between them. Each of which shows unique order of the features, due to the characteristics of the class type. $Data_1$ for example, with classes Benign and Adware, the features were inclined to IAT features which mean the range time (total time between two actions) of the Flow, forward, and backward directions. Further, $Data_2$ classes (Benign, Ransomware) related more to feature such as the total bytes transmitted to the initial window in a forward direction and the duration of two packets transmission in both directions. As for $Data_3$ with classes Benign and Scareware the features closer to features that mean termination of data transmission alongside the time duration and flow of transmission between two packets. On the other hand, $Data_4$ (Benign, SMSmalware) correlates with features like termination of the data, initial window transmission, and minimum segment volume in the forward direction. Finally, $Data_5$ that has the largest number of classes, which are Benign, Ransomware, Scareware, and SMSmalware. The data shows more bonds with features such as transfer time duration of two packets in the forward direction, minimum segment size in the forward direction, transmission duration in the backward direction, and the flow maximum time being idle until it moves again.

7 Conclusion and future work

Android OS has been dominating the market share worldwide in the past few years. The number of users and applications increases every year due to this lead. Therefore, hackers and attackers exploit this success to spread various types of malware. Such issues can be resolve by using a measure like a machine learning Android malware detection-based. Consequently, in this work, we proposed a hybrid Support Vector Machine (SVM) and Harris Hawks Optimization (HHO) approach to detect these malware. The HHO is responsible for two procedures in this approach, optimizing of SVM hyperparameters and features weighting, while the SVM is in charge of evaluating this combination and selecting the best model for the testing phase of CICMalAnal2017 sampled datasets. Furthermore, a detailed analysis of the relationship between the features and malware attacks was presented. The performance of the proposed approach outperforms the other approaches in most datasets and measures. This approach suffers from two main limitations, namely, time consumption and computational complexity. The time consumption limitation can be solved using the correct application and appropriate dataset. While the computational complexity is hard to overcome due to the requirements needed for the objective of this work, parameter optimization and feature weighting, where both require different and unique structure representations. In future work, we seek to investigate more sub-attack types as well as other machine learning methods and metaheuristic algorithms. There are more than twenty sub-attack types that can be reviewed to improve the detection phase. Also, other classification methods can be employed that could offer different results and analyses.

Acknowledgements The research reported in this publication was supported by the Deanship of Scientific Research and Innovation at Al-Balqa Applied University in Jordan (Grant Number: DSR-2020#227).

Author contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by OAA, JAA and AMA-Z. The first draft of the manuscript was written by MAH and UK. All authors commented on previous versions of the manuscript, read and approved the final manuscript.

Funding The research reported in this publication was funded by the Deanship of Scientific Research and Innovation at Al-Balqa Applied University, Al-Salt, Jordan. (Grant Number: DSR-2020#227).

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest. The authors declare that there is no conflict interests regarding the publication of this paper.

Human and animal rights statement This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent statement None.

References

1. ODea, S.: Smartphone users worldwide 2016-2023 (2021). https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
2. Mosa, A.S.M., Yoo, I., Sheets, L.: A systematic review of healthcare applications for smartphones. BMC Med Informat Decision Making **12**(1), 1–31 (2012)
3. Statcounter: Mobile operating system market share worldwide (2021). https://gs.statcounter.com/os-market-share/mobile/worldwide
4. Department, S.R.: Number of apps available in leading app stores as of 4th quarter 2020 (2021). https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/#:$\sim$:text=As
5. Alzaylaee, M.K., Yerima, S.Y., Sezer, S.: Dl-droid: Deep learning based android malware detection using real devices. Computers & Security **89**, 101663 (2020)
6. Dhalaria, M., Gandotra, E.: Android malware detection techniques: A literature review. Recent Patents on Engineering **15**(2), 225–245 (2021)

7. Chen, T.M., Blasco, J., Alzubi, J., Alzubi, O.: Intrusion detection. IET **1**, 1–9 (2014)
8. Wang, X., Li, C.: Android malware detection through machine learning on kernel task structures. Neurocomputing **435**, 126–150 (2021)
9. Agrawal, P., Trivedi, B.: Machine learning classifiers for android malware detection. In: Data Management, Analytics and Innovation, pp. 311–322. Springer (2021)
10. Rajagopal, A.: Incident of the week: Malware infects 25m android phones (2019). https://www.cshub.com/malware/articles/incident-of-the-week-malware-infects-25m-android-phones
11. BBC: One billion android devices at risk of hacking (2020). https://www.bbc.com/news/technology-51751950
12. GOODIN, D.: Google play has been spreading advanced android malware for years (2020). https://arstechnica.com/information-technology/2020/04/sophisticated-android-backdoors-have-been-populating-google-play-for-years/
13. Vaas, L.: Android malware flytrap hijacks facebook accounts (2021). https://threatpost.com/android-malware-flytrap-facebook/168463/
14. Lakshmanan, R.: New android malware uses vnc to spy and steal passwords from victims (2021). https://thehackernews.com/2021/07/new-android-malware-uses-vnc-to-spy-and.html
15. Raveendranath, R., Rajamani, V., Babu, A.J., Datta, S.K.: Android malware attacks and countermeasures: Current and future directions. In: 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 137–143. IEEE (2014)
16. Alqatawna, J., AlaM, A.Z., Hassonah, M.A., Faris, H., et al.: Android botnet detection using machine learning models based on a comprehensive static analysis approach. Journal of Information Security and Applications **58**, 102735 (2021)
17. AlaM, A.Z., Alqatawna, J., Paris, H.: Spam profile detection in social networks based on public features. In: 2017 8th International Conference on information and Communication Systems (ICICS), pp. 130–135. IEEE (2017)
18. Alqatawna, J., Madain, A., AlaM, A.Z., Al-Sayyed, R.: Online social networks security: Threats, attacks, and future directions Social media shaping e-publishing and academia, pp. 121–132. Springer New york (2017)
19. Alzubi, O.A.: A deep learning- based frechet and dirichlet model for intrusion detection in iwsn. Journal of Intelligent & Fuzzy Systems (2021). https://doi.org/10.3233/JIFS-189756
20. Al-Zoubi, A., Alqatawna, J., Faris, H., Hassonah, M.A.: Spam profiles detection on social networks using computational intelligence methods: the effect of the lingual context. Journal of Information Science **47**(1), 58–81 (2021)
21. Al-Ahmad, B., Al-Zoubi, A., Abu Khurma, R., Aljarah, I.: An evolutionary fake news detection method for covid-19 pandemic information. Symmetry **13**(6), 1091 (2021)
22. Alqahtani, E.J., Zagrouba, R., Almuhaideb, A.: A survey on android malware detection techniques using machine learning algorithms. In: 2019 Sixth International Conference on Software Defined Systems (SDS), pp. 110–117. IEEE (2019)
23. Anderson, H.S., Kharkar, A., Filar, B., Roth, P.: Evading machine learning malware detection. Black Hat (2017)
24. BalaGanesh, D., Chakrabarti, A., Midhunchakkaravarthy, D.: Smart devices threats, vulnerabilities and malware detection approaches: a survey. European Journal of Engineering and Technology Research **3**(2), 7–12 (2018)
25. Ma, Z., Ge, H., Liu, Y., Zhao, M., Ma, J.: A combination method for android malware detection based on control flow graphs and machine learning algorithms. IEEE access **7**, 21235–21245 (2019)
26. Rana, M.S., Rahman, S.S.M.M., Sung, A.H.: Evaluation of tree based machine learning classifiers for android malware detection. In International Conference on Computational Collective Intelligence, pp. 377–385. Springer New York (2018)
27. Taheri, R., Javidan, R., Shojafar, M., Vinod, P., Conti, M.: Can machine learning model with static features be fooled: an adversarial machine learning approach. Cluster Computing **23**(4), 3233–3253 (2020)
28. Ananya, A., Aswathy, A., Amal, T., Swathy, P., Vinod, P., Mohammad, S.: Sysdroid: a dynamic ml-based android malware analyzer using system call traces. Cluster Computing pp. 1–20 (2020)
29. Wang, C., Xu, Q., Lin, X., Liu, S.: Research on data mining of permissions mode for android malware detection. Cluster Computing **22**(6), 13337–13350 (2019)
30. Rashidi, B., Fung, C., Bertino, E.: Android malicious application detection using support vector machine and active learning. In: 2017 13th International Conference on Network and Service Management (CNSM), pp. 1–9. IEEE (2017)
31. Sun, J., Yan, K., Liu, X., Yang, C., Fu, Y.: Malware detection on android smartphones using keywords vector and svm. In: 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), pp. 833–838. IEEE (2017)
32. Yang, M., Chen, X., Luo, Y., Zhang, H.: An android malware detection model based on dt-svm. Security and Communication Networks **2020** (2020)
33. Han, H., Lim, S., Suh, K., Park, S., Cho, S.j., Park, M.: Enhanced android malware detection: An svm-based machine learning approach. In: 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 75–81. IEEE (2020)
34. AlaM, A.Z., Faris, H., Alqatawna, J., Hassonah, M.A.: Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts. Knowledge-Based Systems **153**, 91–104 (2018)
35. Alzubi, J.A., Jain, R., Alzubi, O.A., Thareja, A., Upadhyay, Y.: Distracted driver detection using compressed energy efficient convolutional neural network. Journal of Intelligent & Fuzzy Systems (2021). https://doi.org/10.3233/JIFS-189786
36. Vaishanav, L., Chauhan, S., Vaishanav, H., Sankhla, M.S., Kumar, R.: Behavioural analysis of android malware using machine learning. Int. J. Eng. Comput. Sci **6**(5), 21378–21389 (2017)
37. Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W., Ye, H.: Significant permission identification for machine-learning-based android malware detection. IEEE Transactions on Industrial Informatics **14**(7), 3216–3225 (2018)
38. Alzubi, O.A., Alzubi, J.A., Alweshah, M., Qiqieh, I., Al-Shami, S., Ramachandran, M.: An optimal pruning algorithm of classifier ensembles: dynamic programming approach. Neural Computing and Applications **32**, 16091–16107 (2020)
39. Alzubi, O.A., Alzubi, J.A., Tedmori, S., Rashaideh, H., Almomani, O.: Consensus-based combining method for classifier ensembles. The International Arab Journal of Information Technology **15**, 76–86 (2018)
40. Chen, L., Hou, S., Ye, Y.: Securedroid: Enhancing security of machine learning-based detection against adversarial android malware attacks. In: Proceedings of the 33rd Annual Computer Security Applications Conference, pp. 362–372 (2017)
41. Alzaylaee, M.K., Yerima, S.Y., Sezer, S.: Emulator vs real phone: Android malware detection using machine learning. In: Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics, pp. 65–72 (2017)
42. Mahindru, A., Singh, P.: Dynamic permissions based android malware detection using machine learning techniques. In: Proceedings of the 10th innovations in software engineering conference, pp. 202–210 (2017)

43. Wen, L., Yu, H.: An android malware detection system based on machine learning. In: AIP Conference Proceedings, p. 020136. AIP Publishing LLC (2017)
44. Alweshah, M., Alzubi, O.A., Alzubi, J.A., Alaqeel, S.: Solving attribute reduction problem using wrapper genetic programming. International Journal Of Computer Science and Network security **16**, 78–84 (2016)
45. Wang, X., Zhang, D., Su, X., Li, W.: Mlifdect: android malware detection based on parallel machine learning and information fusion. Security and Communication Networks **2017** (2017)
46. Ali, W.: Hybrid intelligent android malware detection using evolving support vector machine based on genetic algorithm and particle swarm optimization. IJCSNS **19**(9), 15 (2019)
47. Faris, H., Habib, M., Almomani, I., Eshtay, M., Aljarah, I.: Optimizing extreme learning machines using chains of salps for efficient android ransomware detection. Applied Sciences **10**(11), 3706 (2020)
48. Manavi, F., Hamzeh, A.: A new approach for malware detection based on evolutionary algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1619–1624 (2019)
49. Hussain, K., Neggaz, N., Zhu, W., Houssein, E.H.: An efficient hybrid sine-cosine harris hawks optimization for low and high-dimensional feature selection. Expert Systems with Applications **176**, 114778 (2021)
50. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (1995)
51. AlaM, A.Z., Heidari, A.A., Habib, M., Faris, H., Aljarah, I., Hassonah, M.A.: Salp chain-based optimization of support vector machines and feature weighting for medical diagnostic information systems. In: Evolutionary Machine Learning Techniques, pp. 11–34. Springer (2020)
52. James, G., Witten, D., Hastie, T., Tibshirani, R.: An introduction to statistical learning, vol. 6, p. 978. Springer, New York (2013)
53. Scholkopf, B., Smola, A.J.: Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press (2001)
54. Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H.: Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems **97**, 849–872 (2019)
55. Lashkari, A.H., Kadir, A.F.A., Taheri, L., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark android malware datasets and classification. In: 2018 International Carnahan Conference on Security Technology (ICCST), pp. 1–7. IEEE (2018)
56. Lashkari, A.H., Kadir, A.F.A., Gonzalez, H., Mbah, K.F., Ghorbani, A.A.: Towards a network-based framework for android malware detection and characterization. In: 2017 15th Annual conference on privacy, security and trust (PST), pp. 233–23309. IEEE (2017)
57. Ideses, I., Neuberger, A.: Adware detection and privacy control in mobile devices. In: 2014 IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEI), pp. 1–5. IEEE (2014)
58. Omeleze, S., Venter, H.S.: Testing the harmonised digital forensic investigation process model-using an android mobile phone. In: 2013 Information Security for South Africa, pp. 1–8. IEEE (2013)
59. Hamandi, K., Chehab, A., Elhajj, I.H., Kayssi, A.: Android sms malware: Vulnerability and mitigation. In: 2013 27th International Conference on Advanced Information Networking and Applications Workshops, pp. 1004–1009. IEEE (2013)

Omar A. Alzubi received his BSc degree from University of Jordan, Jordan. He obtained his MSc degree with distinction in Computer and Network Security from New York Institute of Technology, New York, USA in 2006. In 2013, he received his Ph.D. degree in Computer and Network Security from Swansea University, Swansea, UK. Currently he is an associate professor in the Computer Science Department at Al-Balqa Applied University, Jordan. Dr. Alzubi research interests include computer and network security, machine learning, and cryptography. His cumulative research experience for over ten years resulted in publishing more than thirty articles in highly impacted journals. Dr. Alzubi is the vice dean of Prince Abdullah Bin Ghazi Faculty of Information and Communication Technology. In addition he is an editorial board member and reviewer of many prestigious journals in computer science field.

Jafar A. Alzubi is an associate professor at Al-Balqa Applied University, School of Engineering, Jordan. Received Ph.D. degree in Advanced Telecommunications from Swansea University, Swansea – UK (2012). Master of Science degree (Hons.) in electrical and computer engineering from New York Institute of Technology, New York - USA (2005). And Bachelor of Science degree (Hons.) in Electrical Engineering, majoring in Electronics and Communications, from the University of Engineering and Technology, Lahore – Pakistan (2001). Currently, he is IEEE senior member, and an editorial board member and reviewer in many other prestigious journals in computer science and engineering.

Ala' M. Al-Zoubi received the B.Sc. degree in software engineering from Al-Zaytoonah University, in 2014, and the M.Sc. degree in web intelligence from The University of Jordan, Jordan, in 2017. He is currently pursuing the Ph.D. degree with the School of Science, Technology and Engineering, University of Granada, Granada, Spain. During his graduate studies, he has published several publications in well-recognized journals and conferences. He has worked as a Teacher and a Research Assistant on several projects funded by The University of Jordan. He is a member of two research groups, the JISDF research group that focuses on bridge the gap between the academic and industry mechanisms in security and the Evo-ML Research group, where the group focuses on evolutionary algorithms, machine learning, and their applications for solving important problems in different areas. His research interests include evolutionary computation, machine learning, and security in social network analysis and other research fields.

Mohammad A. Hassonah is a Researcher and a Software Engineer. He completed his Bachelor's degree in Business Information Systems from the University of Jordan, and his Master's degree in Web Intelligence from the same university. His research interests are mainly focused on Machine Learning, Intelligent and Knowledge Systems, and Evolutionary Computation.

Utku Kose received the B.S. degree in 2008 from computer education of Gazi University, Turkey as a faculty valedictorian. He received M.S. degree in 2010 from Afyon Kocatepe University, Turkey in the field of computer and D.S. / Ph. D. degree in 2017 from Selcuk University, Turkey in the field of computer engineering. Between 2009 and 2011, he has worked as a Research Assistant in Afyon Kocatepe University. Following, he has also worked as a Lecturer and Vocational School - Vice Director in Afyon Kocatepe University between 2011 and 2012, as a Lecturer and Research Center Director in Usak University between 2012 and 2017, and as an Assistant Professor in Suleyman Demirel University between 2017 and 2019. Currently, he is an Associate Professor in Suleyman Demirel University, Turkey. He has more than 100 publications including articles, authored and edited books, proceedings, and reports. He is also in editorial boards of many scientific journals and serves as one of the editors of the Biomedical and Robotics Healthcare book series by CRC Press. His research interest includes artificial intelligence, machine ethics, artificial intelligence safety, biomedical applications, optimization, the chaos theory, distance education, e-learning, computer education, and computer science.