Received: March 8, 2024. Revised: April 10, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Feature Selection Using Adaptive Weight Based Grey Wolf Optimization for Malware Detection in Android

Arun Singh Thakur[1]* **Kireet Muppavaram[1]**

[1]*Department of Computer Science and Engineering, GITAM Deemed to be University, Hyderabad, India*
* Corresponding author's Email: arunsingh184@gmail.com

Abstract: The malware application is one of the most dangerous threats to various applications, particularly in Android devices. To effectively detect the malware in Android, the existing researchers utilized the Machine Learning (ML) approaches. However, the researchers have not accurately detected the malware due to the malware complexity, continuous changes as well as increased damages caused by attackers. In this paper, an Adaptive Weight based Grey Wolf Optimization (AWGWO) based feature selection is proposed for malware detection in Android. The proposed AWGWO is evaluated by using Drebin and CICInvesAndMal2019 dataset which contains 216 and 428 malware features. Then, the acquired dataset is pre-processed by utilizing the Min-Max normalization technique to remove the computational complexity and achieve minimum error rates. In the classification stage, a voting combination of benign and malignant malware is used while utilizing three ensemble ML approaches. The ensemble approaches involve a Support Vector Machine (SVM), Random Forest (RF) and AdaBoost algorithm. The proposed AWGWO method achieves better results by using evaluation metrics like accuracy, precision, recall, and F1-score of 0.9953, 0.9930, 0.9963, and 0.9725 respectively which is comparatively better than the existing methods named Random Forest-OWL (RF-OWL), SVM, Adjacency Matrix (AdMat) and Effective Feature selection - RF (EF-RF).

Keywords: Adaptive weight based grey wolf optimization, Android, Machine learning, Malware detection, Support vector machine.

1. Introduction

Malware is a term abbreviated from "Malicious Software", which represents the unnecessary software types unreliable of their purpose, types, and diffusion approach (viruses, spyware, worms, etc.) [1]. Since 2012, Android has been one of the most widely used operating systems (OS). It encompasses a range of concealed malicious activities alongside benign ones, distributed across app stores [2, 3]. Thereby, alarmingly enhancing the rate of malicious applications and their difficulties has become a serious challenge. Every ten seconds, recent reports represent that at least one malware is declared in every app store [4, 5]. This has led to escalating malware attacks between Android devices/applications and OS. The malware is commonly determined by four main groups such as hardware, kernel, hardware abstraction layer as well as application-based attacks [6]. This malware causes a number of material as well as non-material damages named unauthorized action on behalf of users, collection of significant information from users, damaging the device hardware [7].

Many researchers aimed to develop malware detection techniques for detecting malware in Android mobile [8]. The detection of malware in Android is performed by utilizing three various methods such as static, dynamic as well as hybrid. Static determination is an estimation approach employed throughout the application files without the requirement to run the application. The dynamic estimation approach is based on application activity by running it on a virtual or real device [9, 10]. The malware detection performance depends on the selection of a correct set of features. The selected features are provided as input to the classification process to detect the malware, which has a significant effect on malware detection [11, 12]. The existing researchers develop various Machine Learning (ML) for the detection and classification of malware in the android. The performance of the malware identification achieved high accuracy in the identification or classification of the malware by utilizing the ML approaches [13-15]. Even, though many researchers have utilized the ML approaches to detect and solve the problem of malware, the number of irrelevant features affects the system's accuracy and performance. To address these problems, the AWGWO-based feature selection approach is proposed to effectively select relevant malware features in Android. The primary contributions of this research are as follows:

- This research mainly focuses on introducing the detection of malware in Android devices utilizing the AWGWO in the feature selection approach.
- The proposed method selects statistical features and provides to the Ensemble ML approach such as Support Vector Machine (SVM), Random Forest, and AdaBoost approach based on the majority voting for the classification of the malware into benign and malignant.
- The proposed method's effectiveness is calculated by utilizing different assessment metrics like accuracy, precision, recall and F1-score.

The remaining paper is arranged as follows: Section 2 discusses the literature review. Section 3 provides the proposed methodology. Section 4 discusses the experimental results and finally, section 5 discusses the conclusion.

2. Literature survey

Hadeel Alazzam [16] implemented a feature selection-based wrapper method for Android malware detection. A wrapper approach contained the methods of Modified Binary Owl Optimizer (MBO) as well as Random Forest (RF) classifier. An ML algorithm based on ensemble learning of RF was utilized in the process of regression tasks or classification. The MBO accelerated a convergence curve crucially, which affected the performance of the entire model. The suggested method automatically balanced the dataset even though the class was more infrequent than the other classes in the data. However, the multiple trees in RF may cause the system slow as well as inefficient in identifying real-time applications.

Vasileios Syrris and Dimitris Geneiatakis [17] introduced an effective and sufficient ML classification algorithm for the classification of Android malware. The ML algorithms such as Chi-square Test, Bernoulli Naive Bayes (BNB), L1 and L2 regularization, RF, Shallow Neural Network (SNN) as well as SVM were utilized for the classification of the malware. Those algorithms were utilized to estimate the various configurations according to the detection of malware capacity as well as the selection of the feature. The suggested approach was challenging in that the way of method evaluated every feature regarding the consideration of many feature subsets.

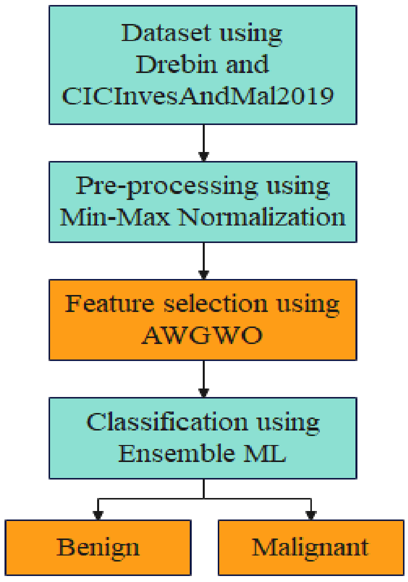
Long Nguyen vu and Souhwan Jung [18] developed to utilized the effective Adjacency Matrix (AdMat) framework to characterize the malware in Android applications. In this system, the matrix acted as "input images" to the Convolutional Neural Network (CNN) and allowed it to learn to classify benign and malicious apps. The AdMat allowed the feature embedding approach to achieve better performance in the analysis of call graphs. The AdMat significantly clarified the images and forwarded them to the quick processing stage. The AdMat was not robust over the dynamic code loading attacks because of the nature of static analysis.

Mohammad Reza Keyvanpour [19] presented the Effective Feature selection - Random Forest (EF-RF) approach-based feature selection for the detection of malware in Android. The suggested approach was utilized for the three various feature selection methods such as maximum weight, effective, as well as effective group selection of the feature, which can be applied in pre-processing. The RF was used to detect selected features of the malware of the Android apps. The suggested approach led to durability by a large number of features as well as interdependency, also vulnerable to reducing the noise data. However multiple trees could make the algorithm slow as well as inefficient for real-time applications.

Durmus Ozkan Sahin [20] developed the ML-based detection of malware to differentiate Android malware from the applications of benign. The Linear Regression (LR) based feature selection approach was utilized to eliminate the unwanted features in the collected data. The feature vector dimension and training time were reduced by utilizing the approach, as well as the classification approach was utilized in the detection of real-time malware applications. The number of permissions needed for ML was minimized by the LR-based attribute approach. However, the suggested approach had consumed the longer time to detect the malware.

Diyana Tehrany Dehkordy and Abbas Rasoolzadegan [21] presented the detection approach to find the malware to enhance accuracy as well as reduction of error rates. The static determination

Received: March 8, 2024. Revised: April 10, 2024.


Dataset using Drebin and CICInvesAndMal2019
Pre-processing using Min-Max Normalization
Feature selection using AWGWO
Classification using Ensemble ML
Benign
Malignant
Figure. 1 Workflow of the suggested method

approach was utilized for the feature extraction of the malware applications as well and the ranking feature approaches were utilized to pre-process the feature set as well and minimum effective features were terminated. The K-nearest Neighbor (KNN), SVM as well as Dichomiser 3 were utilized for the classification to detect models. The suggested approach was most efficient in higher dimensions. However, the highest frequency of features was not appropriate in the development of a malware detection approach.

Ismail Atacak [22] developed the Fuzzy Logic-based Dynamic Ensemble (FL-BDE) approach for the detection malware uncovered to Android OS. The FL-BDE approach involved the structure which integrated the processing power of ML-based approaches as well as decision-making power of the Mamdan-type fuzzy inference Sysetm (FIS). The suggested approach utilized the six ML approaches for the voting and routing-based classification. The developed approach had even performed when data were missing. However, the developed approach's acquired result was much lower than the error rate values than the other approaches except Neural Network (NN).

Amerah Alabrah [23] introduced the automated Artificial Neural Network (ANN)-based detection of malware in Android. The renovated feature sets were provided to the classifier of ANN to categorize the malware into binary classes such as benign (0) and malignant (1). The suggested approach utilized the two benchmark datasets to validate the effectiveness and performed the data preprocessing by static features to normalize the features. The suggested approach was efficient in overall metrics as well as enhanced the robustness of the model. However, the introduced approach had utilized only the minimum robustness.

Santosh K. Smmarwar [24] implemented the Optimized and Efficient Ensemble Learning-based Android Malware Detection, which is known as "OEL-AMD". The suggested approach performed the statistical feature for the reduction of non-informative features. The Binary GWO (BGWO) was utilized for the selection of features to arrange the optimal feature sets. Simultaneously, various learners are trained through hyper-parameter tuning to enhance the induction of cognitive ability of the ensemble approach for classification as well as aggregated estimation. However, high-dimensional features can lead to the classification challenges due to data was calculated as well as tracked at minimum sampling interval.

3. Proposed methodology

In this research, an Adaptive Weight based Grey Wolf Optimization (AWGWO) is proposed for malware detection. The Drebin dataset is utilized in this research which contains 123,453 benign and 5560 Malware applications. The collected dataset is pre-processed by utilizing the Min-Max normalization technique to achieve minimum error rates and forwarded to the feature selection process. The AWGWO is used for the feature selection process and classification is done by voting strategy using ensemble ML approaches. The ensemble approach can classify the malware into benign and malignant based on the majority voting of the classifiers. Fig. 1. represents the workflow of the suggested method.

3.1 Dataset

The Drebin [25] is a publicly available dataset that is utilized to develop the proposed method. This dataset contains 123,453 benign from the app store and 5560 Malware applications. A dataset from the Android apps contains extracting the features that are collected from the Android manifest as well as Dex code. Each Android app has an Android manifest.xml file which explains the required data about an app to the Android development tool as well as support for application installation as well as later execution. The major objection with Drebin was the separate framework of the record and the number of files that must be connected to acquire the entire record with the class labels. The CICInvesAndMal2019 [26] is one of the popular benchmark datasets of malware. This dataset involves 954 and 630 samples in rows with the 8111 columns handled in Excel and a total of 428 sample features. Then collected dataset was provided as input to the pro-processing step.

3.2 Pre-processing

The collected dataset is pre-processed before the feature selection and classification. The collected dataset generally has irrelevant features as well as missing values, which greatly impacts the accuracy of the detection of the malware. So, the data pre-processing technique is significant for the detection and classification processes. Moreover, the pre-processing technique is used to remove the computational complexity and minimum error rates. In this method, the Min-Max normalization pre-processing techniques are utilized to normalize the data in each feature based on the presence of minimum and maximum values in a feature.

3.2.1. Min-max normalization

The Min-Max normalization [27] is the process of pre-processing input data. The data is scaled in the range from 0 to 1 and there is no change in the distribution of data. For every feature, the minimum and maximum values of the feature get the values into 0 and 1. The rescaling process to the $[0, 1]$ interval is completed by exchanging the values of each attribute. The min-max normalization is done before the feature selection and it is expressed in Eq. (1) as;

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

Where, X' – normalization result; X – normalized data; X_{min} and X_{max} – minimum and maximum values from the data of each column. The pre-processed data is forwarded to the feature selection process relevant features selection.

3.3 Feature selection

The pre-processed outcome is considered as input to the feature selection process. It is the process of selecting the most important features from the pre-processed data to enhance the model performance. The feature selection approaches are utilized to select salient features to examine the accuracy in the search space as well as to position the ideal solution. In the feature selection process, the Adaptive GWO algorithm is used and a detailed description of this algorithm is given in a subsequent section. The 216 and 428 malware features from Drebin and CICInvesAndMal2019 are provided as input to the GWO algorithm.

3.3.1. Grey Wolf Optimization

The GWO deceits in hunting grey wolves' actions and public background. This method determines an enhanced approach by employing prey hunting. A strategy to hunt wolves through using multiple phases like prey encircling, prey hunting, and attacking and these methods are used for the determination of optimal approach. In prey encircling, hunting plays a significant part in initializing the prey circle. The optimum approach is determined by the procedure of encircling the grey wolf. The original prey location is analyzed in the present scenario however an optimal solution is not supported in the optimal problem. Therefore, an arbitrary determination of optimum position is analyzed based on solutions like alpha (α), beta (β), and delta (δ). The wolves begin with the attacking procedure by prey as well as the movement of the wolves. The major authority of the alpha is to make decisions such as hunting, sleeping as well as wake-up time. The Beta is familiar with supporting the alpha in making decisions as well as giving input. The Omega is handled by both alpha and beta wolves and it obeys all the wolves.

The Omega wolves are the minimum ranking and they should give in to all wolves and finally eat in a pack. The other wolves in the group named delta (δ), assent to alpha and beta wolves, however, control omega wolves. In GWO, circling behavior is determined in the following Eq. (2) as;

$$\vec{Y}(t+1) = \vec{Y}_p(t) + \vec{A}.\vec{D} \tag{2}$$

Where, $\vec{Y}(t+1)$ – circling behavior of the GWO; $\vec{A}.\vec{D}$ – coefficient vectors; $\vec{Y}_p$ – location vector of the hunter; $\vec{Y}_{wolf}$ – wolves' location in d-dimensional space in which d depicts the number of parameters; t – number of iterations. The mathematical expression of $\vec{D}$ is expressed in Eq. (3) as;

$$\vec{D} = \left|\vec{C}\vec{Y}_p(t) - \vec{Y}(t)\right| \tag{3}$$

Now, $\vec{A}$ and $\vec{C}$ is formulated in Eqs. (4) and (5) as;

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \tag{4}$$

$$\vec{C} = 2.\vec{r}_2 \tag{5}$$

Primarily, number of position candidates are modernized, after that, the wolves' exploitation as well as exploration are utilized to calculate the standard accurate formula. The renewal process of location of grey wolf optimization are presented in Eqs. (6)-(11) as follows;

$$\vec{Y_1} = \vec{Y_\alpha} - A_1.(D_\alpha) \quad (6)$$

$$\vec{Y_2} = \vec{Y_\beta} - A_2.(D_\beta) \quad (7)$$

$$\vec{Y_3} = \vec{Y_\delta} - A_3.(D_\delta) \quad (8)$$

$$D_\alpha = |C_1.Y_\alpha - \vec{Y}| \quad (9)$$

$$D_\beta = |C_2.Y_\beta - \vec{Y}| \quad (10)$$

$$D_\delta = |C_3.Y_\delta - \vec{Y}_{wolf}| \quad (11)$$

The final location of the individual grey wolf is expressed in Eq. (12) as follows;

$$\vec{Y}(t+1) = \frac{\vec{Y_1}+\vec{Y_2}+\vec{Y_3}}{3} \quad (12)$$

Where, C_1, C_2 and C_3 with A_1, A_2 and A_3 - coefficient vectors of wolves. $\vec{Y_1}$, $\vec{Y_2}$ and $\vec{Y_3}$ - best wolf based on individual prey in search space. Y_α, Y_β and Y_δ - position of the present solution and $\vec{Y}$ - inertia constant. GWO has some limitations such as it requires to choice of some maximum iteration value T which impacts the computational time of the optimization. To solve this problem, an adaptive weight-based GWO (AWGWO) is proposed to make the independent optimization algorithm for the number of iterations.

3.3.2. Adaptive weight factor-based GWO

AWGWO develops the strict hierarchical task system of the population of grey wolves by simulating the nature, international mechanism as well as hunting behavior of the grey wolf. Based on (12), the ending position of wolf ω is examined through the average direction and length towards wolves α, β and δ. In the real environment, α, β, and δ wolves in the population are various due to a severe hierarchical procedure of the grey wolf population. Hence, AWGWO is proposed within a GWO's position improvement method. According to the change rate of divergent distance in the prior segment, the solution of the candidate dynamic distribution can be efficiently contemplated as well as Adaptive weight is developed by the change rate of divergent distance. In this process, the location of the wolf is aggressively enhanced by a general convergence degree in an iteration process to efficiently enhance the performance of the optimization approach.

The divergent distance λ majorly contemplated as contiguity among every individual and α wolf. The mathematical calculation of the divergent distance of every α, β and δ is expressed in Eq. (13) as follows;

$$dist_{x\,max} = \max_{i=1,2,..,N}\left(\sqrt{\sum_{d=1}^{D}\left(X_i^d - X_x^d\right)^2}\right) \quad (13)$$

Where, X_i^d and X_x^d – D-dimensional vector of i and x ; $x = \alpha, \beta$ and δ . The change rate of the divergent distance of every individual according to α, β, and δ are formulated in Eqs. (14)-(16) as follows;

$$\lambda_\alpha = \frac{dist_{\alpha\,max} - dist_{\alpha ave}}{dist_{\alpha\,max}} \quad (14)$$

$$\lambda_\beta = \frac{dist_{\beta\,max} - dist_{\beta ave}}{dist_{\beta\,max}} \quad (15)$$

$$\lambda_\delta = \frac{dist_{\delta\,max} - dist_{\delta ave}}{dist_{\delta\,max}} \quad (16)$$

Where, λ_α, λ_β and λ_δ – a variation of divergent distance of individual grey wolves. After, subsequent adaptive weight factors can be developed based on the current divergent distance change rate of every individual concerning α, β, and δ is expressed in Eqs. (17)-(19) as follows;

$$\omega_1 = \frac{\lambda_\alpha}{\lambda_\alpha+\lambda_\beta+\lambda_\delta} \quad (17)$$

$$\omega_2 = \frac{\lambda_\beta}{\lambda_\alpha+\lambda_\beta+\lambda_\delta} \quad (18)$$

$$\omega_3 = \frac{\lambda_\delta}{\lambda_\alpha+\lambda_\beta+\lambda_\delta} \quad (19)$$

Where, ω_1, ω_2 and ω_3 – controls a degree of α, β, and δ on other wolves. Integrated by developed adaptive weight, a new position updated Eq. (20) is acquired as;

$$X_i(t+1) = \frac{\omega_1 X_i\alpha + \omega_2 X_i\,\beta + \omega_3 X_i\,\delta}{3} \quad (20)$$

Where, $X_i\alpha$, $X_i\,\beta$ and $X_i\,\delta$ distance among individual i and wolf α, β and δ. The initial population has been developed arbitrarily, where every solution is a binary vector of 0 and 1, where 0 and 1 denote the absence and presence of a feature. The wolves can be estimated as to fitness function performed in Eq. (21) as.

$$F = W_a * tpr + W_b * \frac{1}{fpr} + W_c * \frac{F_N}{F_S} \quad (21)$$

Where, F – fitness function; W_a, W_b and W_c – weights for true positive, false positive, and selected features ratio. F_N – feature vector length and F_S – number of selected features. In this section, an optimal number of feature sets and relevant features have been efficiently selected by utilizing the AWGWO algorithm. In that, 172 and 342 malware features are selected by using the GWO algorithm. Then, selected features are then provided for the classification process.

3.4 Classification

After the selection of the feature, the selected feature's output is utilized as input for the classification. The ensemble ML is utilized based on three methods for measuring the ranks and weights of the individual classifiers. The optimal subset of the features is selected based on the weights of the majority voting. In the classification process, the Ensemble ML approach such as SVM, RF, and AdaBoost is utilized for the classification of selected malware features in Android as benign and malignant. The detailed description and working of the ensemble approach is provided in the subsequent section

3.4.1. Support Vector Machine

SVM is widely utilized for classification as well as regression problems and it solves the overfitting problem. The SVM handles the non-linearly separable difficulties by converting data by kernel functions. The analysis that is closest to an optimal hyperplane is called a support vector. The aim is to initially categorize the data into binary classes and then mapped into multi-classes. In SVM, there are various hyperplanes, that distinguish them from binary classes which are acquired through determining an optimal hyperplane. SVM refers few training points in the training sample set, that are nearest to the classification decision-making process as well as the complex data points that are classified. While the distance between those points to the classification hyperplanes extends the maximum value, the best classification level in SVM is achieved. For the problem of linear classification, the mathematical formula of the decision classification is stated in Eq. (22) as;

$$f(\vec{x};\ \vec{a}) = sign\left(b + \sum_{j=1}^{m} a_j y_j \langle \vec{x}_j, \vec{x} \rangle\right) \quad (22)$$

Where, b – offset from origin, y_j – labels correlated with samples x ; m – support vector quantity. An inner product $\langle \vec{x}_j, \vec{x} \rangle$ is estimated among the support vectors $\vec{x}_j$ as well as a new vector $\vec{x}$ are to be classified. The support vectors are obtained from the solution of the subsequent optimization issue, which is stated in Eq. (23) as;

$$argmax \sum_{j=1}^{m} a_j - \frac{1}{2} \sum_{j=1}^{m} \sum_{j=1}^{m} a_j a_l y_j y_l \langle \vec{x}_j, \vec{x} \rangle \quad (23)$$

The SVM utilizes the kernel approaches to modify from the raw linear space to a greater one in another dimension for the problems of non-linear. In high-dimensional linear space, the sample is categorized based on the hyperplane. The decision classification after the kernel function introduction is stated in Eq. (24) as;

$$f(\vec{x};\ \vec{a}) = sign\left(b + \sum_{j=1}^{m} a_j y_j K \langle \vec{x}_j, \vec{x} \rangle\right) \quad (24)$$

Where, $K\langle \vec{x}_j, \vec{x} \rangle$ – Kernel function. The hyperparameter is the scaler function, which handles the endurance over the errors. The greater parameter of the hyperparameter leads to the number of difficult decision surfaces, which divides the better number of training vectors whereas the low parameter outcomes in a larger-margin separating hyperplane, depicting the number of endurances to the misclassification. Eventually, the weight vector can be formulated in Eqs. (25) and (26) as follows;

$$\vec{w} = \sum_{j=1}^{m} a_j y_j \vec{x}_j \quad (25)$$

$$b = y_s - \langle \vec{x}, \vec{x}_s \rangle, \forall_s : 1 \le s \le m \ such\ that\ a_s \ne 0 \quad (26)$$

Where, w – weight vector, b – bias; $|b|/\|w\|$ – perpendicular distance to the hyperplane. In this section, the malware has been efficiently and significantly classified by the SVM. The SVM feature vector splits the features into binary classes such as 0 and 1 where 0 represents the benign and 1 represents the malignant. This will make it easier to utilize the DREBIB dataset as well as perform a fair comparison with the various methods. Then, the classified output is then provided to the experimental results process to evaluate the performance of the modal. The advantage of SVM can work better in linear data, is more effective in high dimensions and it is not sensitive to outliers.

3.4.2. Random Forest

The RF is an ensemble ML approach that often utilizes bootstrapping, averaging as well as bagging to train the number of decision trees (DT). By performing the discrete subsets of nearby characteristics, various independent DTs can be developed simultaneously on different segments of the training samples. The RF classification integrates the various DTs for the final decision; as an outcome, the RF classifier has efficient generalization. The RF aims to consistently outperform all other approaches in terms of precision without the problems of imbalanced datasets as well as overfitting. The advantage of RF can solve the overfitting problem and it can efficiently perform even if the data contains null and missing values.

3.4.3. AdaBoost

The AdaBoost approach is an ensemble and iterative boosting approach. In this approach, the predicted outcome of the weak generated by every iteration is compared with an actual outcome. The weight of the current training sample as well as the weight of the current weak learner in the compound predictor are updated based on the variation among predicted and actual outcomes. After iterating and updating the training samples for the further weak learners, the prediction analysis is employed until the final weak learner ends. Hence, this approach significantly combines multiple “weak” learners to create the “strong” prediction for producing accurate predictions. The advantage of the AdaBoost algorithm is minimally prone to overfitting as the input parameters are not jointly optimized and it can be efficiently utilized for the classification of binary problems.

In this classification process, the ensemble approach is utilized for training both weak and strong learning approaches and integrating the results for providing the final prediction. In the training set, the learning approaches are deliberately generated and trained. Later, every instance of the test is estimated on ‘n’ generated approaches as well as predicted class labels are noted. Eventually, the majority voting is utilized to make the final prediction. In the voting process, the predictions from all the classifiers are integrated according to the malware-category-wise prediction. For instance, whether the SVM and RF are performed in voting, SVM predicts benign and RF predicts malignant. In the final prediction process, the benign will be extended if SVM is more accurate in predicting the benign than RF.

4. Experimental results

In this section, an obtained outcome of the proposed method is implemented by using Python 3.8 environment and the system specification by Windows 10 OS, Intel Core i7 processor, and 16GB RAM. The effectiveness of the proposed method is validated by utilizing various assessment metrics named accuracy, precision, recall, False Positive Rate (FPR), and F1-score. The mathematical formula of every metric is described as the following Eqs. (27)-(31) as;

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

$$Precision = \frac{TP}{TP+FP} \tag{28}$$

$$Recall = \frac{TP}{TP+FN} \tag{29}$$

$$F1-score = \frac{2TP}{2TP+FP+FN} \tag{30}$$

$$FPR = \frac{FP}{TN+FP} \tag{31}$$

Where, True Positive (TP) – The malwares that were correctly identified as malwares.

True Negative (TN) - The malware that was correctly identified as benign.

False Positive (FP) - The malware that was incorrectly identified as malware.

False Negative (FN) - The malware that was incorrectly identified as benign.

4.1 Quantitative and qualitative analysis

This section represents the performance analysis of the proposed AWGWO-based SVM approach to various performance metrics. Table 1 depicts the analysis of optimization-based feature selection algorithms. Tables 2 and 3 represent the performance of ML-based classification results after feature selection.

Table 1 and Fig. 2 represent performance analysis of optimization-based feature selection algorithms using Drebin dataset. The performance of Particle Swarm Optimization (PSO), Ant Bee Colony (ABC), Cuckoo Search Optimization (CSO), and GWO are analyzed with AWGWO. The attained outcomes depict that a proposed AWGWO attains better results by performance metrics as accuracy of 0.9728, precision of 0.9689, recall of 0.9678, F1-score of 9626, and FPR of 0.1867.

Table 2 and Fig. 3 represent performance analysis

Table 1. Performance analysis of optimization-based feature selection algorithms

Method	Accuracy	Precision	Recall	F1-score	FPR
PSO	0.8923	0.9075	0.9044	0.8926	0.3065
CSO	0.9178	0.9256	0.9262	0.9154	0.2945
ABC	0.9375	0.9371	0.9482	0.9321	0.1988
GWO	0.9587	0.9478	0.9585	0.9589	0.1956
AWGWO	0.9728	0.9689	0.9678	0.9626	0.1867

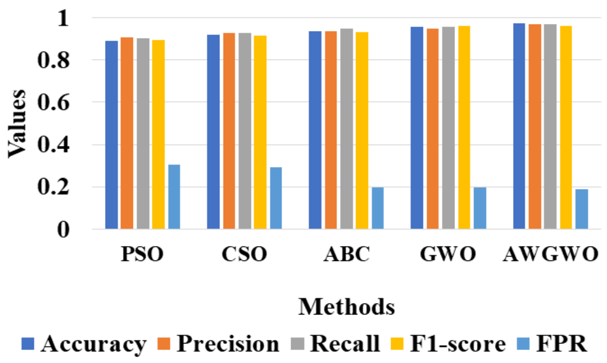


Figure. 2 Graphical representation of the performance of feature selection

of ML-based classification algorithms using Drebin dataset. The effectiveness of the ensemble ML approach is compared with Artificial Neural Network (ANN), RF, AdaBoost and SVM approaches. The attained outcomes depict that the proposed SVM achieves better results by using performance metrics like the accuracy of 0.9856, precision of 0.9735, recall of 0.9789, F1-score of 0.9678 and FPR of 0.1725 respectively.

Table 3 and Fig. 4 depict performance analysis of classification results after the feature selection using Drebin dataset. The performance of AGWO-ANN, AGWO-RF, AGWO-AdaBoost and AGWO-SVM are analyzed with AWGWO-ensemble ML. The attained outcomes depict that a proposed AWGWO-SVM achieves better results by using performance metrics like the accuracy of 0.9953, precision of 0.0030, recall of 0.9963, F1-score of 0.9725, and FPR of 0.0010.

4.2 Comparative analysis

Tables 4 and 5 compare the proposed AGWO-SVM method to a collection of existing algorithms using the Drebin and CICInvesAndMal2019 dataset. The existing methods such as [16-19, 22-24] are compared by using accuracy, precision, recall, F1-score and FPR respectively.

Table 2. Performance analysis of classification results

Method	Accuracy	Precision	Recall	F1-score	FPR
ANN	0.9057	0.9501	0.9147	0.9012	0.2099
RF	0.9219	0.9776	0.9329	0.9590	0.2189
AdaBoost	0.9486	0.9643	0.9577	0.9476	0.1701
SVM	0.9738	0.9588	0.9601	0.9334	0.1843
Ensemble ML	0.9856	0.9735	0.9789	0.9678	0.1725

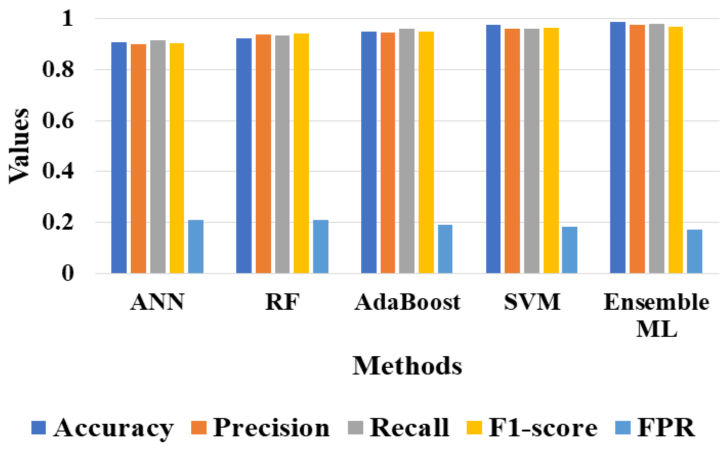


Figure. 3 Graphical representation of performance of classification

Table 3. Performance analysis of classification results after feature selection

Method	**Accuracy**	**Precision**	**Recall**	**F1-score**	**FPR**
AGWO-ANN	0.8829	0.9189	0.8976	0.9056	0.2289
AGWO-RF	0.9273	0.9613	0.9509	0.9428	0.2108
AGWO-AdaBoost	0.9989	0.9568	0.9467	0.9377	0.0926
AGWO-SVM	0.9826	0.9767	0.9828	0.9574	0.1045
AWGWO-ensemble ML	0.9953	0.9930	0.9963	0.9725	0.0010

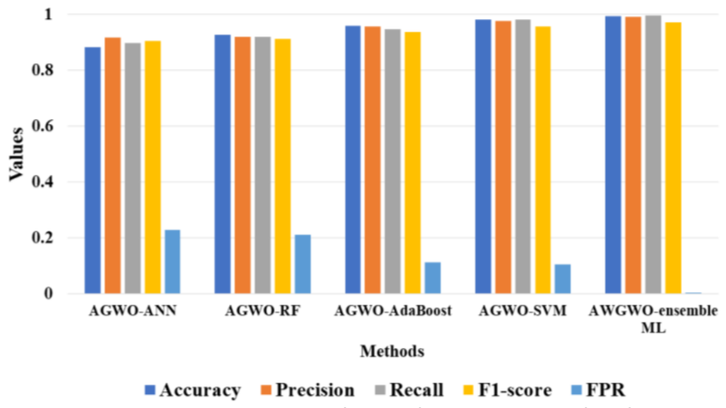


Figure. 4 Graphical representation of classification result after feature selection

Table 4. Comparative Analysis using Drebin dataset

Method	**Accuracy**	**Precision**	**Recall**	**F1-score**	**FPR**
RF-Owl [16]	0.9881	0.9924	0.9956	0.8634	0.1693
SVM [17]	0.997	0.978	0.959	0.969	N/A
AdMat [18]	N/A	0.92	0.92	0.92	N/A
EF-RF [19]	0.9949	N/A	0.92	0.939	0.0013
FL-BDE [22]	0.9933	0.9868	1.00	0.9934	N/A
Proposed AWGWO-ensemble ML	0.9953	0.9930	0.9963	0.9725	0.0010

Table 5. Comparative analysis using CICInvesAndMal2019 dataset

Method	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Layer-based ANN [23]	95.30	95.80	97.60	92.40
BGWO-optimized ensemble [24]	96.166	95.998	94.898	95.989
Proposed AWGWO-ensemble ML	97.383	96.292	98.371	96.227

4.3 Discussion

The previous approaches have some drawbacks like RF-Owl [16] having multiple trees could make the algorithm as slow as well as inefficient for real-time applications. SVM [17] was challenging in a way that evaluating every feature subset is time-consuming. AdMat [18] was not robust over the dynamic code loading attacks because of the nature of static analysis. The proposed AWGWO-based ensemble ML overcomes these drawbacks. The AWGWO is good for both the local exploitation and global exploration phase and it improves the overall convergence speed. Ensemble ML is utilized to improve the capability of the system to detect malware accurately and effectively. The proposed AWGWO-ensemble ML algorithm achieved an accuracy of 0.9953 compared to BGWO respectively.

The proposed AWGWO-ensemble ML achieves the accuracy of 0.9953, precision of 0.9930, recall of 0.9963, F1-score of 0.9725 respectively. However, the existing FL-DDE [22] attained accuracy of 0.9933, precision of 0.9868, recall of 1.00 and F1-score of 0.9934 respectively. The primary objective the proposed method is to enhance the accuracy performance and to avoid the misclassification. The proposed AWGWO-ensemble ML approach achieves the maximum accuracy and precision as compared to the FL-DDE [22] approach. The proposed AWGWO-ensemble ML approach significantly outperforms well and slightly varied as compared to the FL-DDE [22] approach. These results proven that the proposed AWGWO-ensemble ML approach outperforms well and it achieves the superior accuracy results than the existing methods.

The RF-Owl [16] had selected the 205 features from 476 features (44%). The AdMat [18] had 17626 features from the 23000 features (76%). The EF-RF [19] had selected the 11381 features from 123453 (9%). In this proposed method, the AWGWO selected 172 features from the 216 features (80%). As a result, the overall system has obtained better results by performing the selection of relevant features using AWGWO. After the feature selection, the ANN obtained poor results when compared to the other classifiers. Once the feature selection is performed, the better performance is acquired from the AWGWO. In the ensemble approach, the RF achieved better performance as compared to the AdaBoost. As in the overall ensemble ML approach, the SVM performance has enhanced, while the performance of the RF and AdaBoost decreases. As the dimension is minimized through the AWGWO-based feature selection the learning process of this approach is reduced. The advantage of the ensemble ML approach minimized the error-causing factors, thereby ensuring the accuracy as well as stability of the ML approaches.

Thus, the effectiveness, understandability as well as the applicability of the proposed model are certified. As these conditions, it is realized through the number of classifiers that the features that depressingly cause the classification performance are eliminated with the proposed AWGWO-based feature selection method.

5. Conclusion

In this research, a new method Adaptive Weight based Grey Wolf+- Optimization (AWGWO) based feature selection is proposed for the detection of malware in Android devices. Feature selection is a significant process and it is used to eliminate irrelevant features in malware. The android datasets named Drebin and CICInvesAndMal2019 are utilized to estimate the performance of the proposed method. The collected dataset is pre-processed by using the Min-Max normalization technique which is utilized to remove the computational complexity and minimum error rates. The pre-processed dataset is then provided as input to the feature selection process to effectively select the relevant features present in the dataset and it is done by using the AWGWO algorithm. The features are then classified by voting strategy using ensemble ML approaches and it classifies the malware into benign and malignant. The aim of using the ensemble ML approach is to obtain more accurate results in overall metrics. The proposed method achieves better results and it is evaluated by using the various performance metrics. The proposed method achieves the accuracy result of 0.9953 when compared to the existing methods. In the future, the proposed method will cover detecting malware by using hybrid optimization algorithms.

Notation

Variable	**Description**
X'	Normalization result
X	Actual data
X_{min} and X_{max}	Minimum and maximum values from the data of each column
$\vec{Y}(t+1)$	Circling behavior of the GWO
$\vec{A}.\vec{D}$	Coefficient vectors
$\vec{Y}_p$	Location vector of the hunter
$\vec{Y}_{wolf}$	Wolves' location in d-dimensional space
d	Number of parameters
t	Number of iterations
C_1, C_2 and C_3 with A_1, A_2 and A_3	Coefficient vectors of wolves
$\vec{Y}_1$, $\vec{Y}_2$ and $\vec{Y}_3$	Best wolf based on individual prey in search space
Y_α, Y_β and Y_δ	Position of the present solution
$\vec{Y}$	Inertia constant
X_i^d and X_x^d	D-dimensional vector of i
λ_α, λ_β and λ_δ	A variation of divergent distance of individual grey wolve
ω_1, ω_2 and ω_3	Controls a degree of α, β, and δ on other wolves
X_iα, X_i β and X_i δ	Distance among individual i and wolf α, β and δ
F	Fitness function
W_a, W_b and W_c	Weights for true positive, false positive, and selected features ratio
F_N	Feature vector length
F_S	Number of selected features
b	Offset from origin
y_j	Labels correlated with samples x
m	Support vector quantity
$K\langle\vec{x}_j, \vec{x}\rangle$	Kernel function
w	Weight vector
b	Bias

$\lvert b\rvert/\lVert w\rVert$	Perpendicular distance to the hyperplane
FP	False Positive
TN	True Negative
FN	False Negative
TP	True Positive

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, TAS and KM; methodology, TAS; software, KM; validation, KM; formal analysis, KM; investigation, TAS; resources, KM; data curation, TAS; writing—original draft preparation, TAS; writing—review and editing, KM; visualization, TAS; supervision, KM; project administration, KM.

References

[1] Y. Seyfari, and A. Meimandi, “A new approach to android malware detection using fuzzy logic-based simulated annealing and feature selection”, *Multimedia Tools and Applications*, Vol. 83, pp. 10525-10549, 2023.

[2] I.U. Haq, T.A. Khan, and A. Akhunzada, “A dynamic robust DL-based model for android malware detection”, *IEEE Access*, Vol. 9, pp. 74510-74521, 2021.

[3] R. Islam, M.I. Sayed, S. Saha, M.J. Hossain, and M.A. Masud, “Android malware classification using optimum feature selection and ensemble machine learning”, *Internet of Things and Cyber-Physical Systems*, Vol. 3, pp. 100-111, 2023.

[4] J. Singh, D. Thakur, T. Gera, B. Shah, T. Abuhmed, and F. Ali, “Classification and analysis of android malware images using feature fusion technique”, *IEEE Access*, Vol. 9, pp. 90102-90117, 2021.

[5] M. Chaudhary, and A. Masood, “RealMalSol: real-time optimized model for Android malware detection using efficient neural networks and model quantization”, *Neural Computing and Applications*, Vol. 35, No. 15, pp. 11373-11388, 2023.

[6] A. Fournier, F. El Khoury, and S. Pierre, “A client/server malware detection model based on machine learning for android devices”, *IoT*, Vol. 2, No. 3, pp.355-374, 2021.

[7] E. Odat, and Q.M. Yaseen, “A Novel Machine Learning Approach for Android Malware Detection Based on the Co-Existence of Features”, *IEEE Access*, Vol. 11, pp. 15471-15484, 2023.

[8] İ. Atacak, “An Ensemble Approach Based on Fuzzy Logic Using Machine Learning Classifiers for Android Malware Detection”, *Applied Sciences*, Vol. 13, No. 3, p. 1484, 2023.

[9] I. Almomani, A. Alkhayer, and W. El-Shafai, “An automated vision-based deep learning model for efficient detection of android malware attacks”, *IEEE Access*, Vol. 10, pp. 2700-2720, 2022.

[10] J. Lee, H. Jang, S. Ha, and Y. Yoon, “Android malware detection using machine learning with feature selection based on the genetic algorithm”, *Mathematics*, Vol. 9, No. 21, p. 2813, 2021.

[11] D.V. Nguyen, G.L. Nguyen, T.T. Nguyen, A.H. Ngo, and G.T. Pham, “Minad: Multi-inputs neural network based on application structure for android malware detection”, *Peer-to-Peer Networking and Applications*, Vol. 15, pp. 163-177, 2022.

[12] 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, 2021.

[13] X. Wang, and C. Li, “Android malware detection through machine learning on kernel task structures”, *Neurocomputing*, Vol. 435, pp. 126-150, 2021.

[14] D.Ö. Şahın, S. Akleylek, and E. Kiliç, “LinRegDroid: Detection of Android malware using multiple linear regression models-based classifiers”, *IEEE Access*, Vol. 10, pp. 14246-14259, 2022.

[15] D.Ö. Şahin, O.E. Kural, S. Akleylek, and E. Kılıç, “A novel Android malware detection system: adaption of filter-based feature selection methods”, *Journal of Ambient Intelligence and Humanized Computing*, Vol. 14, pp. 1243-1257, 2023.

[16] H. Alazzam, A. Al-Adwan, O. Abualghanam, E. A. Alhenawi, and A. Alsmady, “An Improved Binary Owl Feature Selection in the Context of Android Malware Detection”, *Computers*, Vol. 11, No. 12, p. 173, 2022.

[17] V. Syrris, and D. Geneiatakis, “On machine learning effectiveness for malware detection in Android OS using static analysis data”, *Journal of Information Security and Applications*, Vol. 59, p. 102794, 2021.

[18] L.N. Vu, and S. Jung, “AdMat: A CNN-on-matrix approach to Android malware detection

and classification”, *IEEE Access*, Vol. 9, pp. 39680-39694, 2021.

[19] M.R. Keyvanpour, M. Barani Shirzad, and F. Heydarian, “Android malware detection applying feature selection techniques and machine learning”, *Multimedia Tools and Applications*, Vol. 82, No. 6, pp. 9517-9531, 2023.

[20] D.Ö. Şahin, O.E. Kural, S. Akleylek, and E. Kılıç, “A novel permission-based Android malware detection system using feature selection based on linear regression”, *Neural Computing and Applications*, Vol. 35, pp. 4903-4918, 2023.

[21] D.T. Dehkordy, and A. Rasoolzadegan, “A new machine learning-based method for android malware detection on imbalanced dataset”, *Multimedia Tools and Applications*, Vol. 80, pp. 24533-24554, 2021.

[22] İ. Atacak, “An ensemble approach based on fuzzy logic using machine learning classifiers for Android malware detection”, *Applied Sciences*, Vol. 13, No. 3, p.1484, 2023.

[23] A. Alabrah, “A Novel Neural Network Architecture Using Automated Correlated Feature Layer to Detect Android Malware Applications”, *Mathematics*, Vol. 11, No. 20, p. 4242, 2023.

[24] S.K. Smmarwar, G.P. Gupta, S. Kumar, and P. Kumar, “An optimized and efficient android malware detection framework for future sustainable computing”, *Sustainable Energy Technologies and Assessments*, Vol. 54, p. 102852, 2022.

[25] Drebin dataset link: https://www.kaggle.com/datasets/shashwatwork/android-malware-dataset-for-machine-learning (Accessed on 02/11/2023).

[26] CICInvesAndMal2019 dataset link: https://www.unb.ca/cic/datasets/invesandmal2019.html.

[27] R. Zhao, J. Yin, Z. Xue, G. Gui, B. Adebisi, T. Ohtsuki, H. Gacanin, and H. Sari, “An efficient intrusion detection method based on dynamic autoencoder”, *IEEE Wireless Communications Letters*, Vol. 10, No. 8, pp. 1707-1711, 2021.