Received: April 11, 2022. Revised: June 6, 2022.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

MH-DLdroid: A Meta-Heuristic and Deep Learning-Based Hybrid Approach for Android Malware Detection

Ravi Mohan Sharma[1]* **Chaitanya P Agrawal[1]**

[1]*Department of Computer Science and Applications, Makhanlal Chaturvedi University, Bhopal, M. P. 462001 India*
* Corresponding author's Email: vision20rm@gmail.com

Abstract: With the fast advancement of smartphone technology, the smartphone has emerged as the most prevailing instrument for accessing the Internet and obtaining a wide range of services with a click. Increased use of smartphones for online payments has attracted fraudsters, adding to an increase in malware outbreaks. Mobile application vulnerabilities and malware are the origins of various types of fraud and numerous cyber-attacks. Large datasets are frequently used for malware analysis; however, large datasets may contain many redundant, inappropriate, and noisy features, resulting in misclassification and low detection rates. This paper presents a hybrid approach to Android malware detection that reduces the dimensionality of datasets to reduce resource-intensive computation while preserving critical information. We present a novel hybrid approach for detecting Android malware based on a meta-heuristic (modified Intelligent Water Drop Algorithm (IWD)) and Deep Learning (DL) techniques. The studies show that the proposed approach efficiently removes irrelevant attributes and attains significant detection performance with an F1-Score of 93.7%, a precision of 95.35, an accuracy of 99.12%, and a recall rate of 96.68%.

Keywords: Deep learning, IWD, Android malware detection, Meta-heuristic methods.

1. Introduction

Smartphones are prevalent nowadays because of their multifunctional capabilities. Smartphones are pervasive in our daily lives, and they're used for everything from online surfing to e-banking, e-learning, purchasing, and social media applications. Android has risen to popularity as a leading mobile operating system in recent decades, with a 73 % market share in June 2021 [1]. Depending on the requirements and purpose, mobile applications can be downloaded from various sources. Malware and benign ware are the two types of Android apps. Malware infects mobile devices and performs a variety of fraudulent activities. In the previous decade, malware has risen at an uncontrollable rate. According to AV-Test, Trojans were responsible for 93.93 % of malware infections, and ransomware came in second [2]. Malware detection has generally depended on signature-based techniques; it derives malware signatures from the source code of Apps. Signature-based detection has several drawbacks, including the inability to detect new malware and the requirement for code to generate signatures. An attacker can conceal the malicious payload as an executable APK/JAR within APK resources. After installing the app, this malware loads the DexClassLoader API and executes the dynamic code. Malware can trick a user into installing an embedded APK by posing as a critical update. As a result, malware detection based on behavior is becoming more widespread. The proposed work is based on behavior-based malware detection.

In behavior-based analysis, a large number of attributes are retrieved from APK files. As a result, a comprehensive dataset is created, which may contain numerous duplicates and unusable and noisy features. Analyzing large datasets requires a large amount of memory, computational power, and time. So we need an appropriate algorithm to select highly efficient features from the dataset. The feature selection

strategy removes elements that are either irrelevant or will have little or no impact on the result. Many nature-inspired meta-heuristics algorithms, such as the grey wolf optimizer (GWO), genetic algorithm(GA), and bat algorithm, have demonstrated their efficiency in feature selection in various domains. And deep learning (DL) also achieves the desired results by automatically deducing the features and fine-tuning the optimal features. This research implements a hybrid approach that combines both techniques' best elements. The main offerings of the planned work are as follows

- We offer a novel hybrid approach based on a swarm-based Meta-heuristic algorithm and a deep learning method for malware prediction.
- First, we modified the node selection mechanism of the original IWD, as shown in section 3.1 in step 2, by using the mutual information score instead of the probability function.
- The adapted version of IWD analyzed malware's behavior pattern and minimized the search space. And deep neural network (DNN) is utilized for subset evaluation using higher-level features extraction.
- We examine the outcomes and compare them to previous work to validate the proposed approach. To evaluate the suggested technique, we employed the widely used datasets DREBIN, MALGENONE, and MSGHIC.

The rest of the paper is deliberate as follows: section 2 is dedicated to previous work, section 3 explains the proposed MH-DLdroid, section 4 defines the working process of the proposed hybrid model, and section 5 discusses the datasets and their pre-processing steps and experimental environment, section 6 establishes the performance assessment metrics, section 7 displays the experimental results, section 8. provide the comparison with the existing methods, and section 9 delivers a summary of proposed work with future direction.

2. Related work

This section analyzes the previous functions of malware detection along with their limitations.

In the paper, H. Gao et al. [3] presented a 'GDroid' approach for malware detection based on graph convolutional network (GCN) with a 97 percent accuracy. However, its presentation degrades as the number of real-world samples increases. In another paper, R. Feng et al. [4] introduced the 'MobiTive' approach based on GRU/LSTM and Bidirectional (GRU/LSTM) and achieved an accuracy of 96.75%. M. Cai et al. [5] proposed a function-call graph-based system (E-FCGs) for learning behavior level features representing app runtime behaviors. They investigated the proposed method's performance using LR, DT, SVM, KNN, RF, MLP, and CNN classifiers. The proposed method achieves satisfactory performance but suffers from low dimension datasets. S. K. Sasidharan et al. [6] presented a 'ProDroid' approach for malware detection based on suspicious API classes; this method achieved an accuracy of 94.5 %. This method suffers from a high false alarm rate. In another study, S. Millar et al. [7] presented a multi-view deep learning approach for malware detection with no specialist malware domain insight to select or rank input features. Its reported F1 measure is 99.63. This method suffer from high computational costs in a high-dimensional dataset because there is no mechanism to select optimal features. In another study, T. Lu et al. [8] presented a hybrid DL model based on DBN, GRU, and BPN networks. The stated accuracy of this work is 96.82%. This method suffers from a high computational cost. J. Feng et al. [12] proposed a two-layer malware prediction method. The first layer is a fully connected neural network (FCNN) that applies permission, intent, and component attributes. The second layer is a CNN with an autoencoder to detect malware. N. Zhang et al. [9] proposed TC-Droid, a text classification-based malware detection method that feeds the text sequence of Apps to CNN to explore important information. The model reports an accuracy of 96.6 percent, a precision of 94.6 percent, and F1- score of 96.6 percent, and a recall of 98.4 percent. The main disadvantage of this approach is that it is time-consuming. Arvind Mahindru et al. [10] demonstrated an ML-based approach called 'FSDroid,' built by combining the LSSVM with RBF by employing ten feature selection techniques; and achieved 99% accuracy. The main limitation of this approach is it uses several statistical methods for feature selection. In another paper, T. Kim et al. [11] proposed a multimode neural network (MNN) method for analyzing the VirusShare and Malgenome datasets and reported 98% accuracy. Y. Yang et al. [12] developed the 'DGCNDroid' method, which generates a function call graph that is fed into the deep graph convolutional network. This method's reported detection accuracy is 98.2 percent. This method has poor performance in handling reflection or dynamic payloads. Xi Xiao et al. [13] trained two LSTM network models using the system call sequence as input. It trained one LSTM with malware and the other with benign apps and obtained two similarity scores to identify apps, with a 96.6 percent

accuracy. This model suffers from a high false alarm rate. In another paper, Y. Hei et al.[14] presented 'HAWK,' a malware detection method based on a heterogeneous information network (HIN). The reported accuracy and F1-score for this model are 96.95 % and 96.89 %, respectively, and the detection time is 3.5ms. This model suffers from a high false alarm rate. F. Ou et al. [15] presented an 'S3Feature' approach for malware detection that is based on three types of sensitive graphs (a)sensitive function call graph (SFCG), (b)sensitive subgraphs (SSGs), and (c)sensitive neighbor subgraphs) (NSGs). This model's reported F1-score is 97.71 percent F1-score. The primary disadvantage of this model is that it is ineffective on dynamic payloads. Wenhui Zhang et al.[16] demonstrated a method that combines the XML file's visual features with the DEX file's data section and extracts the images fed to the temporal convolution network (TCN). This model's reported accuracy is 95.44 percent, its recall rate is 95.45 percent, its F1-Score is 95.44 percent, and its precision is 95.45 percent. This model is significantly less sensitive to the dynamic payload. Tatiana Frenklach et al. [17] presented a static malware detection method based on an app similarity graph (ASG). In balanced settings, they reported an accuracy of 97.5 percent and an AUC score of 98.7 percent. The drawback of this technique is that it necessitates a large amount of storage to store the entire set of apps. Long Nguyen Vu et al. [18] formed the 'AdMat,' which generates an adjacency matrix for each application, and these matrices serve as "input images" to the CNN model. The reported accuracy of this model is 98.26 percent, and the F1-score, precision, and recall rate are all 97 percent. This model has a high computation burden and is not resistant to dynamic payloads. P. Xu et al. [19] presented the 'Falcon' architecture, representing network packets as 2D images and fed into a bidirectional LSTM to investigate distinctive attributes. This method has a stated accuracy of 97.16 percent. However, it has a high computational cost because the images demand a lot of calculation. In another paper, M. R. Norouzian et al. [20] presented a 'Hybroid' framework that exhibited 97.0 percent accuracy by using program code structures as static behavioral features and network traffic as dynamic behavioral data. This approach has a dynamic payload problem because it uses a computer code structure. A. Mahindru et al. [21] developed a 'SemiDroid' that used unsupervised machine learning techniques and used authorization and API requests as features vectors to obtain a detection rate of 98.8%. R. Surendran et al. [22] propose a hybrid approach for malware detection based on tree augmented naive bayes (TAN) that retains conditional dependencies between static and dynamic features while achieving 97 percent accuracy. X. Jiang et al. [23] proposed the 'FDP' approach based on fine-grained dangerous permission. They tested 1700 benign and 1600 malicious apps and discovered that FDP has a TP rate of 94.5 percent. This method generates a high number of false alarms. In another study, H. Bai et al. [24] resented a 'FAMD' framework in which they extracted permissions and Dalvik opcode sequences using symmetrical uncertainty to differentiate malware and benign ware using CatBoost classifiers and achieved 97.40 percent accuracy. R. Taheri et al. [25] presented a Hamming distance-based approach for detecting app similarities. The results show that the proposed algorithms have more than 90% accuracy. In some cases (for example, when considering API features), accuracy exceeds 99 percent. Y. Ding et al. [26] presented a CNN-based bytecode image-based malware detection method. Because CNN learns the pattern from the bytecode image, this model avoids selecting optimal attributes from the dataset. By considering apps bytecode, this model suffers from a dynamic payload problem. A. Arora et al. [27] developed the 'PermPair' approach, which identifies the pair of dangerous permissions that are collectively responsible for malware threats. The detection accuracy of the proposed system was 95.44 percent. H Zhu et al. [28] proposed a SEDMDroid framework based on MLP and SVM classifiers, with PCA for feature selection. The permission-sensitive API-based SEDMDroid achieves an accuracy of 89.07 percent. It achieved an accuracy of 94.92 percent by using sensitive data flow information as the attributes. The high false alarm rate is a major disadvantage of this model.

We proposed a model that will reduce the feature space by intelligently selecting significant features using IWD and DNN-based models to overcome the problem of false alarms and classification time. The proposed study focuses solely on malware behavior patterns that will help in avoiding dynamic payload issues.

3. Proposed MH-DLdroid approach

This section is parted into two sections, the first section defines the feature selection process using modified IWD, and the second section describes the applied deep learning technique.

3.1 The modified IWD

In 2009, Shah-Hosseini and Hamed were the first to propose the IWD [29]. This algorithm simulates the behavior of water droplets to follow the most

efficient routes from source to destination and bypass the environmental constraints to create a shorter path. It uses artificially generated intelligent water droplets (IWD) and ambient parameters to choose the best route. The problem is signified by a graph G (N, E), where N indicates the graph's nodes and E denotes the graph's edges. Each water drop gradually travels between the nodes until it reaches its complete solution. An iteration is completed when all IWDs have achieved the final destination, and all subsequent iterations are finished at the maximum value of iteration (I_M).

Step 1. Initialization of static and dynamic parameters

The static parameters were unaffected during the whole process. The artificially created IWD is denoted by N^D, which is distributed over the graph G(N, E), and also it represents the total number of attributes in the dataset or total node in the graph. The velocity of IWDs is updated using three variables V_a, V_b, and V_c; the soil values of the local path are updated using three variables S_a, S_b, and S_c. The I_M represents the total number of iterations taken from the user and I_s denotes the initial soil value of the local pathway. The dynamic parameters are initialized at the beginning of the process and updated. The list of nodes visited by each water drop is initially blank and updated if a particular IWD visits the node. The initial velocity of IWD is denoted by, and initial soil is laden on a drop denoted by S_l set to zero. The values of static and dynamic parameters are shown in Table 1.

Step 2. Modified node selection process

The proposed amendment is implemented in this phase of the original IWD algorithm, and all other steps remain the same. The mutual information score is used in place of the probability function to select the next node in the graph. It computes the mutual information value for each independent variable concerning the dependent variable and picks the ones with the most significant information gain. So it can be a better option to choose the next node in IWD if a water drop D is currently in node i and intends to move j where a node represents a feature. Then mutual information score is calculated using Eq. (1).

$$P_m = \sum_x \sum_y P_{xy}\ (X,Y)\ log \frac{P_{xy(X,Y)}}{P_{x(X)} P_{y(Y)}} \quad (1)$$

The P_m represents the mutual information score of variable X in concern to variable Y, the ($P_{xy(X,Y)}$) represents the joint probability distribution, ($P_{x(X)}$) and ($P_{y(Y)}$) indicates the marginal probability distribution. This function provides the node mutual information, a positive float value between 0 and 1.

Step 3. Update velocity and soil values

Eq. (2) is used to update the velocity ($I^{V(t+1)}$). Eq. (3) denotes the soil value in the local path. Where ρ_r is a constant its range is between 0 and 1. Eqs. (4), (5), and (6) are used to update soil values.

Where (H_D (i, j)) represents the heuristic desirability degree and Eq. (6) represents the time function that is defined as the time required for water drop k to travel from i to j at the time (t+1).

$$I^{V(t+1)} = I^{V(t)} + \frac{v_a}{v_b + v_c * S(i,j)} \quad (2)$$

$$S(i,j) = (1-\rho_{ir}) * (S(i,j)) - \rho_r \Delta S(i,j) \quad (3)$$

$$I^S = I^s + \Delta s \quad (4)$$

$$\Delta S(i,j) = \frac{S_a}{S_b + S_c * time\left(i,j: I^{V(t+1)}\right)} \quad (5)$$

$$time\left(i,j: I^{V(t+1)}\right) = \frac{H_D(i,j)}{I^{V(t+1)}} \quad (6)$$

Step 4. Reinforcement and termination process

The iteration best solution I^{IB} is calculated using Eq. (7). Where I^p denotes all solutions of an iteration, and q(x) defines the fitness function used to evaluate the quality of the solution. The soil of all edges in $I^{IB}$is calculated using Eq.(8).

$$I^{IB} = argmax_{\forall xp \in I^p}\ q(x) \quad (7)$$

$$S(i,j) = (1+\rho_i) * S(i,j) - \rho_i * S_{IB}^k * \left(\frac{1}{q(I^{IB})}\right) \quad (8)$$

$$I^{GB} = \begin{cases} I^{GB} & if\ q(I^{GB}) \geq q(I^{IB}) \\ I^{IB} & otherwise \end{cases} \quad (9)$$

Where ∀ (i, j) $\in I^{IB}$ and ρ_i represents a small constant, The (S_{IB}^k) denotes the k^{th} iteration's soil value in iteration best path. The global best solution $I^{GB}$is calculated as follows. Where Eq. (9) is used to substitute the I^{GB} with I^{IB} or preserve the same value,

Table 1. Static and Dynamic parameters of modified IWD.

Static Parameter	Values	Dynamic Parameters	Values
N^D	(215 or 357)	N^V list of visited node	initially empty {}
V_a, V_b,&V_c	1, 0 .01 , 1		
S_a,S_b,& S_c.	1, 0.01 , 1		
ρ_r , ρ_i	0.9, 0.9, 0.01	I^V	4
I_M	20	I^S	0
I_S	100		

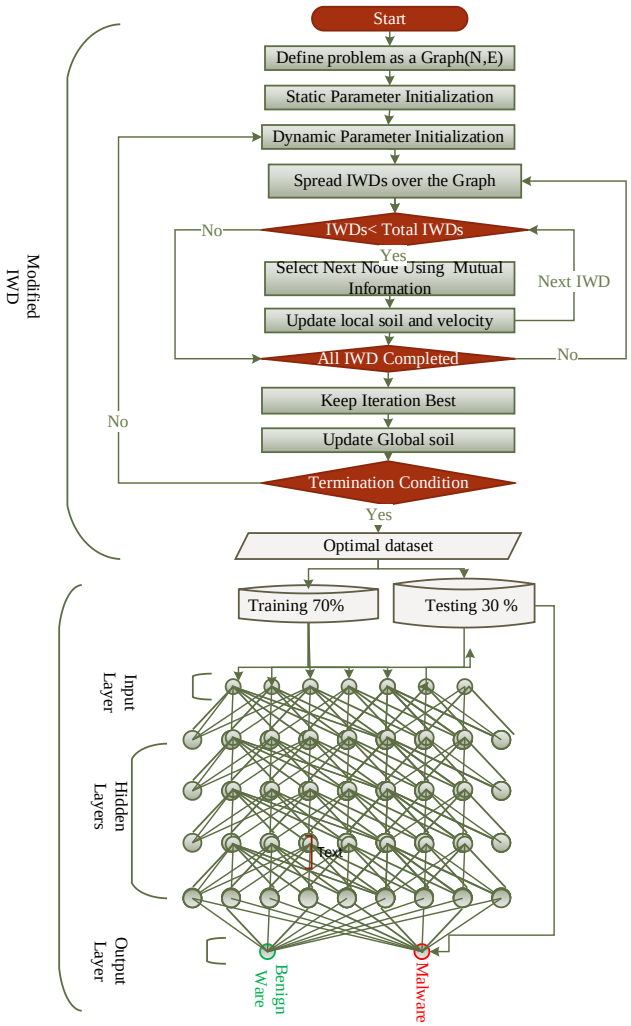

Figure. 1 The flow chart of the proposed MH-DLdroid

the solution construction and reinforcement steps are iterated until the termination state is reached. If the value of I_C (Iteration count) becomes equal to or greater than (Iteration max) I_M, the iteration is stopped.

3.2 Deep neural networks

Deep learning is a subfield of artificial intelligence and machine learning technology that mimic specific human brain functions to make good decisions to intimidate humans and their actions. It's a critical data science component that streamlines predictive modeling and statistics based on data-driven approaches. The proposed DNN is developed in Python and compiled with Keras, a scientific computing platform. The suggested hybrid model is illustrated in Fig. 1, which utilized the modified IWD algorithm for feature selection and the Deep neural network (DNN) for subset evaluation. The various components of suggested DNN models are described below.

- Keras: Keras is a powerful open-source neural network toolkit developed in Python for building deep learning models. It's based on well-known deep learning frameworks like TensorFlow and Theano. Keras is a TensorFlow user interface that includes dropout, batch normalization, pooling layers, objectives, activation functions, and optimizers.
- ReLU: ReLU stands for "Reduced Rectified Linear Activation Function," a rectified network with hidden layers that use the rectifier function. It is a piecewise linear function; if the input X is positive, the ReLU produces X; otherwise, the output is 0.
- Sigmoid: The sigmoid function is also recognized as a squashing function (0, 1) since it compresses the entire number line into a small range, such as 0 and 1. The sigmoid function distinguishes between malware and benign ware in our model.
- ADAM: The ADAM optimizer employs the adam algorithm, which uses the stochastic gradient descent method to carry out the optimization process. We have used the ADAM default parameters, such as alpha (α) 0.001, beta (β1) 0.9, and beta (β2) 0. 999, and epsilon(ε) 10e8.
- Binary cross-entropy (BCE): The binary cross-entropy, also known as log loss, it provide the assessment of the success of a classification model whose output is between 0 and 1. The BCE increases as the anticipated likelihood differ from the actual label.

Table 2. DNN Parameters and Neuron Layer Architecture

DNN Parameters		DREBIN		
Parameters	Values	L#	Shape	P#
Batch Size	20	1	None, 40	2120
Epoch	30	2	None, 12	492
Optimizer	ADAM	3	None, 8	104
Error	BCE	4	None, 1	9
Activation 1	ReLu	Total-Parameters =2725		
Activation 2	Sigmoid			

MALGENOME			MSGHIC		
L #	Shape	P#	L#	Shape	P#
1	None, 62	4526	1	None, 62	13206
2	None, 26	1638	2	None, 30	1890
3	None, 4	108	3	None, 14	434
4	None, 1	5	4	None, 4	60
			5	None, 1	5
Total-Parameter = 6,277			Total-parameters =15595		

- The number of neurons: A neuron is a machine that makes decisions or categorizes something based on specified criteria. The neuron performs some calculations and sends output through a synapse to neurons deep in the neural net. Each synapse has a weighting that influences the relative relevance of the last neuron in the overall neural network. The neuron's activation function receives a weighted total of these signals, and the result is passed to the network's next layer. The neuron layer architecture and DNN parameters are shown in Table 2.

4. Working process of proposed MH-DLdroid

In the first stage, the optimal features are selected using a modified IWD algorithm. The modified IWD examines behavior patterns of android apps and determines the best subset N from the entire dataset M. A fully connected undirected graph G (N, E) defines the optimization problem, where N denotes nodes and E denotes edges. The selection of next node is guided by the mutual information score in the undirected graph. The soil on the edge indicates the obstacles in the local path. Each water drop is distributed randomly over the graph and acts as a searching agent. The result acquired from iteration best solution I^{IB} is used to decide the global best solution I^{GB}. The optimal route is the one with the fewest obstacles. All nodes in the intended optimal path represent the optimal feature subset. The main steps of hybrid mode MH-DLdroid are shown in Algorithm 1.

Algorithm 1. The main setps of MH-DLdroid

1. Input: Feature set of (DREBIN and MALGENOME and MSGHIC Dataset)
2. Output: Optimal feature of all the Three dataset
3. Express problem as a graph G(N,E)
4. Initialization of Static parameters
5. **While (**$I^C < I^M$) , **do**
6. Dynamic parameters initialization
7. Generate and spread artificial IWDs randomly over the graph G(N,E).
8. Update the list of the visited node N_l^V
9. **while** (I < IWD) && (solution not complete**)**, **do**
10. **for** k = 1 to IWD **do**
11. if drop k is in node I
12. And intended for node j then calculate feature importance for j if it is not present in N_l^V
13. Use mutual information score from sklearn to calculate rank of next node
14. Use p_{mi} to move water drop from i to node j .
15. Variable Update velocity I^V of the drop k, Soil within the drop k $\Delta s(i,j)$, Soil within the edge $S_{lv(i,j)}$
16. **End for**
17. **End while**
18. Select the best iteration (I^{IB})
19. update soil value in local edges $S_{lv(i,j)}$ included in the
20. update the (I^{GB}) (global best solution)
21. if (quality of (I^{GB}) < (quality of I^{IB})
22. I^{GB} = I^{IB} （Swap the values)
23. **End while**
24. return (I^{GB} final solution)
25. Remove unwanted features from the datasets
26. Split dataset as training and testing set
27. Apply Sequential DNN
28. Get final outcome

5. Datasets, pre-processing, and experimental environment

In the proposed work, three Android datasets have been taken to do the investigation. The first dataset is a DREBIN-215, which contains 9476 benign and 5560 malware samples from the DREBIN project, mainly consisting of a total of 215 features of API calls, permissions, intents, command signatures, etc. [30]. The MALGENOME is a second dataset that contains 3799 app samples from the Genome project, including 2539 benign-ware samples and 1260 malware. It includes 215 features of API calls, permissions, and intents [31]. The third dataset is MSGHIC, which contains 3090 benign and 3090 malware samples and 357 attributes of API calls, permissions, intents, and command signatures [32]. Duplicate occurrences are deleted from the dataset during the pre-processing of the dataset. The entry with a NaN value is also removed from the dataset.

6. Performance assessment metrics

The confusion matrix represents classification results; it provides insight into classifier performance and reveals which classes are correctly identified and

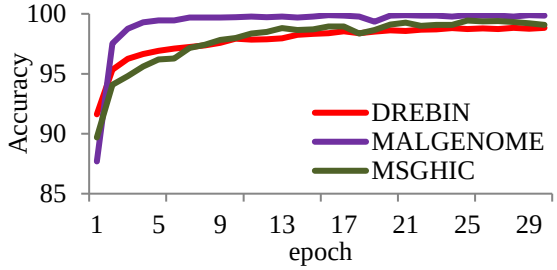

Figure. 2 Training accuracy

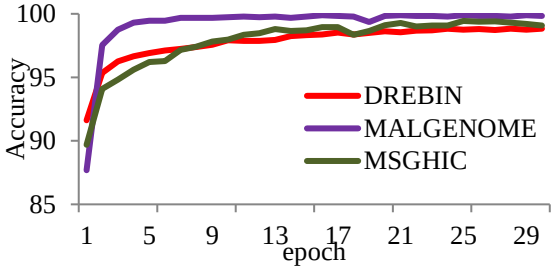

Figure. 3 Testing accuracy

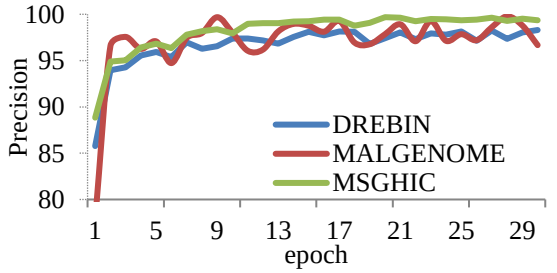

Figure. 4 Training precision

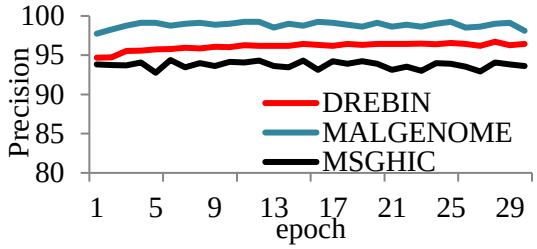

Figure. 5 Testing precision

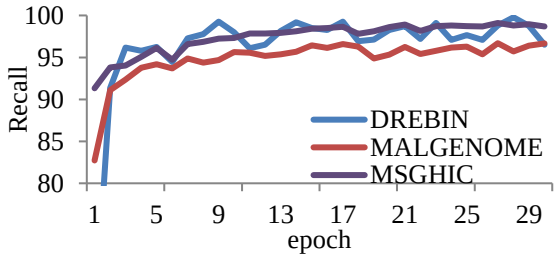

Figure. 6 Training recall

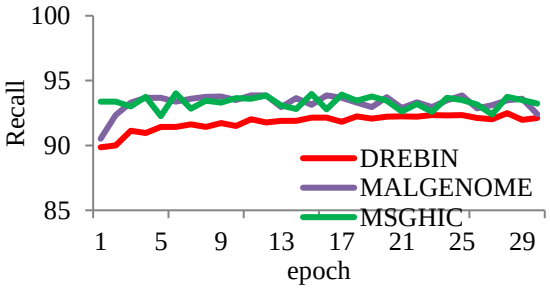

Figure. 7 Testing recall

Table 3. Performance assessment metrics

Assessment Metrics	Formulas
Accuracy(ι)	$\iota=(\gamma+\sigma)/(\mu+\rho+\gamma+\sigma)$
Recall (δ)	$\delta=\gamma/(\gamma+\rho)$
Precision (λ)	$\lambda=\gamma/(\gamma+\mu)$
F1-Score (τ)	$\tau=2(\lambda*\delta)/(\lambda+\delta)$

which are not. Table 3 displays the performance evaluation metrics.

- True positive (γ): A true positive is an occurrence in malware samples that was successfully predicted.
- False positive (μ): A test result indicates that a mobile device has malware while it does not contain malware.
- True negative (σ): A true negative accurately predicts benign-ware in samples.
- False negative (ρ): A test result that indicates that the mobile does not contain malware while the mobile does indeed contain malware.

7. Results and discussions

This section demonstrates the performance of the MH-DLdroid. The modified IWD provides more exploration and exploitation to prevent local optima in search space. As a result, the final feature subsets chosen have high discriminative power. In order to accurately identify the effectiveness of the proposed method, five evaluation parameters are evaluated.

Table 4 displays the training and testing results for the last five epochs derived from the proposed method. The results obtained from the model are shown based on training and testing. The test result represents the trained model identifying independent data not used for training purposes. The training outcome in each performance matrix represents the use of the same data for both training and testing. The suggested model has attained the highest accuracy of 99.12 in malware detection on the MALGENOME dataset and the highest accuracy of 99.8 during training. Fig. 2 and 3. show the obtained accuracy from subset evaluation. It is clear from the obtained

Table 4. The training and testing performance of the last five epochs of DNN

Data set	Epoch #	Training Performance BCE	ևu	τ	ʎ	δ	Testing /Detection Performance BCE	ևu%	τ	ʎ	δ
DREBIN	E-26	0.0407	98.62	96.4	97.43	96.04	**0.167**	96.64	92.8	94.78	92.22
	E-27	0. 0373	98.82	97.2	98.20	96.8	0.1742	96.68	**93.05**	93.86	**93.45**
	E-28	0.0366	**98.86**	97.2	98.36	96.75	0. 1653	**96.71**	93.0	**95.35**	91.96
	E-29	0.0369	98.85	96.8	97.8	96.4	0.1768	96.49	92.98	94.47	92.78
	E-30	**0.0353**	98.86	**97.6**	**98.63**	**97.2**	0.177	96.45	92.57	93.65	92.96
MALGENOME	E-26	0.0225	99.84	97.1	97.2	97.07	0.0542	98.5	92.85	92.9	93.44
	E-27	0.0202	99.84	98.6	98.61	98.79	0.0539	98.62	93.1	93.31	93.44
	E-28	0.0185	99.78	**99.7**	**99.78**	**99.78**	**0.0501**	99	93.47	94.02	93.44
	E-29	**0.0173**	**99.89**	98.7	98.79	98.75	0.0519	**99.12**	**93.61**	**94.27**	**93.44**
	E- 30	0.017	99.84	96.5	96.67	96.53	0.051	98.12	92.42	92.13	93.44
MSGHIC	E-26	0. 0325	98.68	98.6	99.21	98.35	0.4508	93.76	93.39	91.51	**96.68**
	E-27	0. 0261	99.07	98.9	99.54	98.62	**0.3719**	94.22	93.69	94.18	94.24
	E-28	0. 0307	98.98	98.5	99.09	98.25	0.4284	94.07	93.39	92.55	95.58
	E-29	0. 0286	99.04	98.9	99.4	98.69	0.3757	93.84	93.28	94.93	92.87
	E-30	**0. 0194**	**99.44**	**99.3**	**99.95**	**98.95**	0.3886	**94.3**	**93.70**	**94.95**	93.63

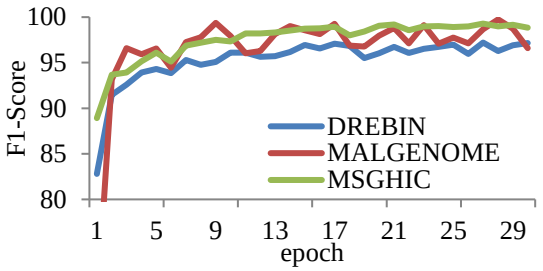

Figure. 8 Training F1- score

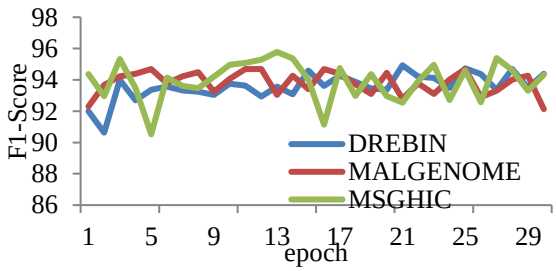

Figure. 9 Testing F1-score

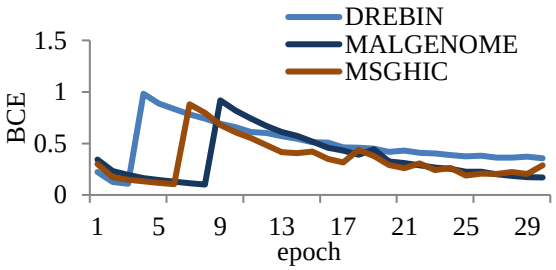

Figure. 10 Training BEC

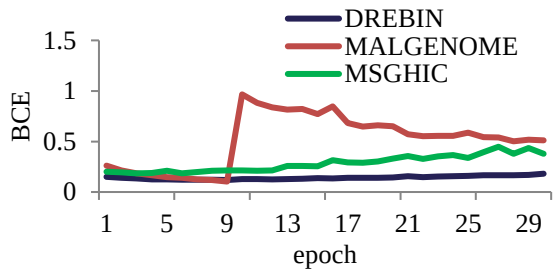

Figure. 1 Testing BEC

results that the subset evaluation of the MALGENOME dataset has achieved the highest precision of 95.35% in malware detection. And 99.78 % precision has been achieved during model training. Fig. 4 and 5. show the precision of training and testing of the model. During malware detection, the model achieved a 96.68 % recall rate on the MSGHIC dataset. And recall rate of 99.78 % has been gained on the MALGENOME dataset during model training. Fig. 6 and 7. show the recall trends of the MH-DLdroid model. The F1-score represents the harmonic mean of precision and recall [33]. The proposed method yielded the highest F1 score of 93.70 percent during malware detection. And the highest F1 score of 99.7 percent was obtained while training the model on the MALGENOME dataset.

Fig. 8 and 9. Depict the F1-Score of the optimal subset evaluation, where the x-axis represents the epoch number, and the y-axis represents the obtained value of F1-scores. Fig. 10 and 11. Illustrate the obtained binary cross-entropy, where the x-axis denotes the epoch, and the y-axis represents the BCE. The model has performed admirably in error rate, the error decreasing as the epoch progresses.

Table 5. The Comparision of MH-DLdroid with some existing approaches.

Reference	**ևւ**	**τ**	**ʎ**	***δ***	**BCE**
Proposed	**99.12**	**93.7**	**95.35**	**96.68**	0.047
Gdroid [3]	98.99	98.9	98.6	98.3	----
MobiTive [4]	96.75	-	96.78	96.72	----
FSDroid [10]	99	96	98	-	----
MNN [11]	98	99	98	99	----

8. The comparison with existing methods

The comparison of the performance of the anticipated approach with some existing methods is given in Table 5. The accuracy of the proposed method is higher than that of other similar methods, but its F1 score, precision, and recall are lower than others. The findings also demonstrate that combining deep learning with meta-heuristic techniques increases performance. Since meta-heuristic algorithms aim to achieve intelligently optimal results by continuously learning from their prior experience, they do not re-evaluate paths that have already been traversed. In real-world problems, the search space is often unknown and complex, and contains a large number of local optima. The stochastic nature of meta-heuristics offers immense possibilities to effectively avoid local optima, comprehensively search the entire search space, and obtain optimal solutions.

Similarly, deep learning incrementally extracts high-level features from data to find the best possible solution [34]. Therefore, in this hybrid approach, the meta-heuristic optimization reduces the learning time and complexity by selecting the appropriate features. Deep neural networks are capable of performing many complex processes at once. Their ability to learn from errors allows them to check the correctness of their predictions and make the necessary corrections. Therefore the resulting hybrid model gives better results; it reduces the size of the search space by more than 60%.

9. Conclusion and future work

In this work, the modified meta-heuristic technique (IWD) is used to intelligently select the prime features with high discriminative power from the dataset, which reduces the size of all datasets by more than 60%. The stochastic nature of meta-heuristics enables them to avoid local optima stagnation and search the entire search space, making them more effective for feature optimization. And deep neural networks (DNN) detect hidden patterns by using abstraction and non-linear transformations of raw input data from unlabeled optimal sets. A thorough examination of the proposed MH-DLdroid model yields significant detection results with a 99.12 % accuracy. And 93.7% of F1-score, a precision of 95.35 %, and a recall rate of 96.68%. This method achieves a very low false alarm rate of 0.88 and a very low BCE of 0.050. This technique is immune to dynamic payload issues because it only looks at app behavior patterns. In future work, we will investigate other behavior patterns like API call patterns and another metaheuristic approach to improve other parameters such as precision, recall, and F1-score in malware detection.

Conflicts of interest

"The authors declare no conflict of interest."

Author contributions

Conceptualization R. M. Sharma, Chaitanya P Agrawal and Kamal Upriti; Methodology R. M. Sharma; Chaitanya P Agrawal; Kamal and Kamal Upriti; writing—original draft preparation, resources data curation, R. M. Sharma; Chaitanya P Agrawal: and Kamal Upriti; review and editing R. M. Sharma; Chaitanya P Agrawal: and Kamal Upriti; All authors have read and agreed to the published version of the manuscript.

References

[1] "Mobile OS market share 2021 | Statista", https://www.statista.com/statistics/272698/.

[2] "Malware Statistics & Trends Report | AV-TEST", https://www.av-test.org/en/statistics/malware/.

[3] H. Gao, S. Cheng, and W. Zhang, "GDroid: Android malware detection and classification with graph convolutional network", *Computers & Security*, Vol. 106, pp. 1-14, 2021.

[4] R. Feng, S. Chen, X. Xie, G. Meng, and S. Lin, Y. Liu, "A performance-sensitive malware detection system using deep learning on mobile devices", *IEEE Transactions on Information Forensics and Security*, Vol. 16, pp. 1563-1578, 2020.

[5] M. Cai, Y. Jiang, C. Gao, H. Li, and W. Yuan, "Learning features from enhanced function call graphs for Android malware detection", *Neurocomputing*, Vol. 423, pp. 301–307, Jan. 2021.

[6] S. K. Sasidharan, and C. Thomas, "ProDroid—An Android malware detection framework based on profile hidden Markov model", *Pervasive and Mobile Computing*, Vol. 72, pp. 1-16, 2021.

[7] S. Millar, N. McLaughlin, J. M. D. Rincon, and P. Miller, "Multi-view deep learning for zero-day Android malware detection", *Journal of Information Security and Application*, Vol. 58, pp. 1-14, 2021.

[8] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wan, "Android malware detection based on a hybrid deep learning model", *Security and Communication Networks*, Vol. 2020, pp. 1-11, 2020.

[9] N. Zhang, Y. Tan, C. Yang, and Y. ZhangLi, "Deep learning feature exploration for android malware detection", *Applied Soft Computing*, Vol. 102, pp.1-7, 2021.

[10] A. Mahindru and A. L. Sangal, "FSDroid:-A feature selection technique to detect malware from Android using Machine Learning Techniques", *Multimedia Tools and Applications*, Vol. 80, No. 9, pp. 13271-13323, 2021.

[11] T. Kim, B. Kang, M. Rho, S. Seze, and E. Im, "A multimodal deep learning method for android malware detection using various features", *IEEE Transactions on Information Forensics and Security*, Vol. 14, No. 3, pp. 773-788, 2019.

[12] Y. Yang, X. Du, Z. Yang, X. Liu, and Yang, et al., "Android malware detection based on structural features of the function call graph", *Electronics*, Vol. 10, No. 2, pp. 1-17, 2021.

[13] X. Xiao, S. Zhang, F. Mercaldo, G.Hu, and A. K. Sangaiah, "Android malware detection based on system call sequences and LSTM", *Multimedia Tools and Applications*, Vol. 78 No. 4, pp. 3979-3999, 2019.

[14] Y. Hei, R. Yang, H. Peng, L. Wan, X. Xu, J. Liu, H. Liu, J. Xu, and L. Sun, "Hawk: Rapid android malware detection through heterogeneous graph attention networks", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 2021, pp. 1-15, 2021.

[15] F. Ou, and J. Xu, "S3Feature: A static sensitive subgraph-based feature for android malware detection", *Computers & Security*, Vol. 112, p. 102513, 2022.

[16] W. Zhang, N. Luktarhan, C. Ding, and B. Lu, "Android malware detection using tcn with bytecode image", *Symmetry*, Vol. 13, No. 7, p. 1107, 2021.

[17] T. Frenklach, D. Cohen, A. Shabtai, and R. Puzis, "Android malware detection via an app similarity graph", *Computers & Security*, Vol. 109, p. 102386, 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] P. Xu, C. Eckert, and A. Zarras, "Falcon: malware detection and categorization with network traffic images", In: *Proc. of International Conf. on Artificial Neural Networks*, Bratislava, Slovaki, pp. 117–128, 2021.

[20] M. R. Norouzian, P. Xu, C, Eckert, and A. Zarras, "Hybroid: Toward Android Malware Detection and Categorization with Program Code and Network Traffic", In*: Proc. of International Conf. on Information Security*, pp. 259–278, 2021.

[21] A. Mahindru and A. L. Sangal, "SemiDroid: a behavioral malware detector based on unsupervised machine learning techniques using feature selection approaches", *International Journal of Machine Learning and Cybernetics*, Vol. 12, No. 5,pp. 1369-1411, 2021.

[22] R. Surendran, T. Thomas, and S. Emmanuel, "A TAN based hybrid model for android malware detection", *Journal of Information Security and Applications*, Vol. 54, p. 102483, 2020.

[23] X. Jiang, B. Mao, J. Guan, and X Huang, "Android malware detection using fine-grained features", *Scientific Programming*, Vol. 2020, No. 5190138, pp. 1-13, 2020.

[24] H. Bai, N, Xie, X. Di, and Q. Ye, "Famd: A fast multifeature android malware detection framework, design, and implementation", *IEEE Access*, Vol. 8, pp. 194729-194740, 2020.

[25] R. Taheri, M. Ghahramani, R. Javidan, M. Shojafar, Z. Pooranian, and M. Contic, "Similarity-based Android malware detection using Hamming distance of static binary features", *Future Generation Computer Systems*, Vol. 105, pp. 230-247, 2020.

[26] Y. Ding, X. Zhang, J. Hu, and W. Xu, "Android malware detection method based on bytecode image", *Journal of Ambient Intelligence and Humanized Computing*, Vol. 2020, pp. 1-10, 2020.

[27] A. Arora, S. K. Peddoju, and M. Conti, "Permpair: Android malware detection using permission pairs", *IEEE Transactions on Information Forensics and Security*, Vol. 15, pp. 1968–1982, 2020.

[28] H. Zhu, Y. Li, R. Li, J. Li, Z. You, and H. Song, "SEDMDroid: An Enhanced Stacking Ensemble Framework for Android Malware Detection", *IEEE Transactions on Network Science and Engineering*, Vol. 8, No. 2, pp. 984–994, 2021.

[29] H. S. Hosseini, "The intelligent water drops algorithm: a nature-inspired swarm-based

optimization algorithm", *International Journal of Bio-inspired Computation*, Vol. 1, Nos. 1-2, pp. 71–79, 2009.
[30] DREBIN-dataset https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_2/
[31] MALGENOME-Dataset https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_1/5854590/
[32] MSGHIC-dataset https://github.com/MSGHIC/Deep_learning_for_android_malware_detection/blob/master/
[33] V. Kumar "Evaluation of computationally intelligent techniques for breast cancer diagnosis", *Neural Computing and Applications*, Vol. 33, No. 8, pp. 3195-3208, 2021.
[34] X. Pei, L. Yu, and S. Tian, "AMalNet: A deep learning framework based on graph convolutional networks for malware detection", *Computers & Security*, Vol. 93, p. 101792, 2020.