Received: May 5, 2025. Revised: July 13, 2025.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Android Malware Detection Using a Modified Dwarf Mongoose Algorithm

Rawan Alabdallat[1]* **Mosleh Abualhaj[1]** **Ahmad Abu-Shareha[2]**

[1]*Department of Networks and Cybersecurity, Al-Ahliyya Amman University, Amman, Jordan*
[2]*Department of Data Science and Artificial Intelligence, Al-Ahliyya Amman University, Amman, Jordan*
* Corresponding author's Email: alabdallatrawan80@gmail.com

Abstract: The ubiquity of mobile devices due to the advancement of Internet technology has increased the threat of Android malware, which places users' data, identities, and finances at risk of theft and cybercriminal activities. Android malware detection (AMD), which uses machine learning, faces challenges with high-dimensional datasets, leading to increased computation and reduced accuracy. Feature selection methods extract key features and eliminate redundant data but are still susceptible to overlooking important features, leading to false positives or negatives. This study aims to enhance AMD via a modified dwarf mongoose optimization algorithm (DMOA) for efficient feature selection. The DMOA is improved with adaptive mechanisms, including crossover and mutation strategies, to explore the solution space better. These enhancements help the DMOA avoid local optima and identify superior feature subsets for more accurate malware detection. The CICAndMal2017 dataset is used to evaluate the effectiveness of different machine learning classifiers. The experimental results demonstrate that the proposed modified DMOA model achieves exceptional accuracy. The decision tree (DT), random forest (RF), and XGBoost models have 100% accuracy in binary classification. Furthermore, the proposed modified DMOA consistently outperforms several existing AMD models — including Particle Swarm Optimization (PSO), Multi-Verse Optimizer (MVO), Enhanced Moth Flame Optimizer (EMFO), Aquila Optimizer, and Grey Wolf Optimizer (GWO)— in terms of accuracy, based on results reported in previous studies using the same dataset.

Keywords: Feature selection, Malware, Malware detection, Machine learning, Dwarf mongoose optimization algorithm.

1. Introduction

The rapid development of the Internet has increased the global use of mobile devices. As of 2022, the number of mobile network smartphone subscriptions has reached 6.4 billion [1]. Approximately 70.93% of smartphone users use the Android operating system [2]. The advancements in Android applications, especially banking, e-commerce, social sites, education, and entertainment, have led Android to be a primary target of malware developers. Due to their open-source nature and compatibility with devices, Android applications are particularly vulnerable to malware attacks [3]. Malware is defined as malicious code that disrupts and damages computer systems and the digital infrastructure. They also damage mobile/electronic devices, such as viruses, worms, trojans, backdoors, ransomware, adware, and spyware. malware aims to harm digital systems and steal valuable information, thereby threatening system confidentiality, integrity, and availability [4]. Malware attacks can also result in serious consequences for businesses and organizations, such as stealing information, interfering with business processes, or damaging reputation. On the basis of a Statista report, the number of malware strains detected and spread worldwide in 2023 reached 6.06 billion. A large portion of these attacks are concentrated in the Asia-Pacific region [5].

Several defense methods, mainly categorized into signature- and behavior-based methods [6], have been developed to detect Android malware. Machine learning algorithms that use well-established classification algorithms typically classify Android

applications as benign or malicious applications [7]. Machine learning has encountered problems in malware detection because of the high dimensionality of the data, which may increase the computation time and reduce accuracy. The feature selection process plays an important role in addressing such concerns. Irrelevant and redundant data can also be removed, enabling more efficient and accurate learning models to detect malware effectively [8]. Feature selection can be classified into two main methods: wrapper methods and filter methods. The wrapper-based method is the most popular method for problem classification. Finding the optimal feature subsets via a wrapper method is difficult because it obtains only the smallest number of subsets despite its high accuracy. Feature selection poses challenges that result from the increased time needed to determine optimal features in a high-dimensional dataset [9].

AMD remains a critical challenge due to the rapid growth of Android users and applications (Sabbah et al., 2022). Although ML methods are widely used (Nawshin et al., 2020), they often struggle with high-dimensional datasets, causing longer computation times and less accurate results (PER, 2021). Feature selection helps reduce irrelevant data (Sangal & Verma, 2020), but can still produce false positives and negatives (Akinola et al., 2022). The evolving nature of malware requires continuous adaptation and new optimization techniques to handle large datasets effectively and improve detection accuracy. Traditional optimization algorithms often get stuck in local optima, selecting poor feature subsets that limit performance (Game & Vaze, 2020). Therefore, advanced optimization methods are needed to efficiently explore the search space and enhance AMD systems without falling into local optima.

Metaheuristic algorithms are widely used for data classification and optimization due to their simplicity, flexibility, and independence of derivatives, enabling them to avoid local optima [10]. A local optimum in optimization problems refers to a state where the objective function's value is optimal, "either at a minimum or maximum," within a limited range of nearby potential solutions [11]. Metaheuristic algorithms are used to solve feature selection problems, especially in high-dimensional datasets, in which the best feature subset should be selected promptly. Algorithms such as the firefly algorithm, particle swarm optimization (PSO), Harris hawks optimization (HHO), and the dwarf mongoose optimization algorithm (DMOA), which are used to find optimal feature subsets, can balance exploration and exploitation to find optimal or near-optimal feature subsets [9]. These algorithms thoroughly investigate the promising search space during the exploration phase. Local searches are conducted during the exploitation phase to find potential locations identified during exploration [10].

In this paper, a modified version of the DMOA is used for feature selection to enhance the performance of the AMD system. The modified DMOA method is assessed via the CICAndMal2017 dataset, which is frequently used to evaluate malware models. On the basis of the nature of foraging in dwarf mongooses, DMOAs maintain a balance between exploitation and exploration. This capability enables DMOAs to effectively discover subsets of features in high-dimensional malware data [12].

The rest of the paper is organized as follows: Section 2 discusses related works. Section 3 presents the proposed model. Section 4 analyses the results. Section 5 concludes the findings and provides recommendations for future research.

2. Related works

This section discusses several previous studies that have been used ML algorithms to detect and categorize Android malware. Some researchers have proposed combining nature-inspired wrapper-based metaheuristics and machine-learning approaches to detect malware on Android devices.

Hossain et al. [13] proposed a novel method for Android ransomware detection via PSO algorithms for traffic characteristic selection with DT and RF classifiers for data traffic classification. The CICAndMal2017 dataset was used and preprocessed via normalization and oversampling to address class imbalance. PSO-based feature selection was used to optimize the detection accuracy. Their proposed method achieved accuracy in detecting ransomware attacks (81.58%).

Taher et al. [14] proposed a hybrid model for detecting and classifying Android malware called DroidDetectMW. Feature selection was performed via metaheuristic optimization algorithms, including MVO and enhanced moth flame optimization (EMFO), and the most relevant features for classification were identified. The detection phase uses ML algorithms, such as RF and SVM, to categorize the Android app's behavior. The CICAndMal2017 was used in the experiment. Significant improvements in precision, recall, and classification accuracy, especially in binary malware classification (98.1% accuracy) and malware category classification (96.9% accuracy), were achieved.

Grace and Sughasiny [15] solved the growing threat of malicious Android programs by using an

Aquila optimizer and a hybrid LSTM-SVM classifier to detect malware. The method was evaluated on the CICAndMal2017 dataset and achieved superior performance results in terms of accuracy (0.97), precision (0.94), recall (0.90), and F1_score (0.93).

Aldehim et al. [16] proposed Gauss-mapping black widow optimization with the DL-enabled Android malware classification (GBWODL-AMC) method for AMD. GBWO was used for feature selection, inspired by the movement of the black widow spider. The deep extreme learning machine (DELM) model was used for malware classification, and the ant lion optimization (ALO) algorithm was used to optimally select the model's parameters. The CICAndMal2017 dataset was used to test the simulation analysis of the GBWODL-AMC approach. The testing results revealed that the GBWODL-AMC technique outperformed other malware detectors (with a maximum accuracy of 98.59% in binary classification, and 98.50% in multiclass classification).

Alissa et al. [17] proposed a DWOML-RWD method that combines dwarf mongoose optimization with ML-driven ransomware detection. The DWOML-RWD model selects features via the enhanced krill herd optimization (EKHO) algorithm, which employs dynamic oppositional-based learning (QOBL). The method uses DWO with an extreme learning machine (ELM) classifier to improve the efficiency of ransomware detection. The DWOML-RWD model was evaluated on a dataset of 840 samples. The results revealed high accuracy, sensitivity, and specificity of 99.40% and false positive and false negative rates of 0.60%.

PER [18] proposed a new technique known as the bat optimization algorithm for wrapper-based feature selection (BOAWFS) to classify application permissions with AMD. BOAWFS is a wrapper classifier inspired by the echolocation principle employed by bats to improve feature selection by reducing the elevation dimension or noise in every high-dimensional dataset. The method was tested on the CICInvesAndMal2019 dataset, which comprises both malware and benign applications. The BOWFS-DT model achieved an accuracy of 93.73%, and malware detection was enhanced.

Guendouz and Amine [19] proposed a novel method called BDA-FS, which uses ML and the dragonfly algorithm for feature selection to detect Android malware. The features were extracted on the basis of a dataset of 5000 malicious applications and 5000 benign applications. The features, which included the Android permissions, were transformed into feature vectors. The dragonfly algorithm was used to select the most pertinent features. The RF, DT, naïve Bayes, SVM, and KNN classifiers were used for classification. The proposed BDA-FS method effectively detected Android malware. The highest accuracy was 96.33% for the random forest classifier.

Smmarwar et al. [20] proposed a new AMD method called OEL-AMD. The BGWO algorithm was employed for feature selection and provided optimal feature sets for the static and dynamic layers. The proposed method was evaluated on the CICInvesAndMal2019 dataset. The best classification accuracy was 96.95% for binary classification utilizing static layer features and 83.49% for multiclass classification utilizing dynamic layer features. The results indicate the effectiveness of the OEL-AMD method for AMD.

Naick et al. [21] proposed an AMD method with swarm intelligence optimization algorithms (BES and SFO). ML classifiers were used to analyze application programming interface (API) calls. The optimal features of the API calls were determined to increase the detection accuracy. The BES and SFO were used for feature selection. ML classifiers, which included the DT, KNN, LR, SVM, and RF classifiers, were used to evaluate the classification performance. The method was tested on the API call sequence dataset. The model achieved an accuracy of 98.92%.

Prasanna and Krishna [22] developed a feature selection method that could differentiate between malware and benign applications on the basis of API calls. Wrapper-based swarm intelligence algorithms were used to optimize the feature selection process involving ALO and the WOA. Various ML methods, such as SVM, RF, DT, GB, KNN, and LR, can distinguish between malicious and benign features. The RF classifier outperformed all the other classifiers studied, with accuracies of 94.04% for the WOA and 91.42% for ALO.

Despite the success of recent studies on AMD, several issues warrant further exploration. Redundant identification of information must be reduced while maintaining high accuracy. For example, although the GBWODL-AMC strategy yields promising results with high accuracy, sensitivity, and specificity when it is applied to the CICAndMal2017 dataset, notable challenges still exist. Combining GBWO for feature selection, DELM for classification, and ALO for parameter optimization increases the level of complexity in the system. However, this hybrid approach may result in high computation and processing times, making it less suitable for real-time applications. Additionally, as the aforementioned approach is good at minimizing the feature set, multiple algorithms tend to complicate integration and optimization procedures. This issue further affects the overall efficiency and scalability of the

Table 1. Summary of metaheuristic optimization algorithms in related works

Ref#	Authors (Year)	Feature Selection Algorithm	Dataset	Results	Limitations
Ref[13]	Hossain et al. (2022)	PSO	CICAndMal2017	Accuracy: 81.58%.	Focused only on ransomware
Ref[14]	Taher et al. (2023)	MVO, EMFO	CICAndMal2017	Binary Classification: Accuracy 98.1%, Multiclass Classification: Accuracy 96.9%.	High computational complexity due to hybrid and multi-phase model
Ref[15]	Grace & Sughasiny (2022)	Aquila Optimizer	CICAndMal2017	Accuracy: 97%, Precision: 94%, TPR: 90%, F1 Score:93%.	High computational complexity
Ref[16]	Aldehim et al. (2023)	GBWO	CICAndMal2017	Binary Classification Accuracy: 98.59%. Multiclass Classification Accuracy: 98.50%.	High computational cost and longer training time
Ref[17]	Alissa et al. (2022)	EKHO, DOBL	840 samples (specific dataset not mentioned)	Accuracy: 99.40%.	High computational complexity
Ref[18]	PER (2021)	BOAWFS	CICInvesAndMal2019	Accuracy: 93.73%.	High computational cost and complexity.
Ref[19]	Guendouz & Amine (2023)	Dragonfly Algorithm	5000 malicious and 5000 benign apps	Accuracy: 96.33% .	Limited Dataset
Ref[20]	Smmarwar et al. (2022)	BGWO	CICInvesAndMal2019	Binary Classification Accuracy: 96.95%, Multiclass Classification Accuracy: 83.49%.	Requires a large dataset for effective training.
Ref[21]	Naick et al. (2022)	BES, SFO	API Call Sequence dataset	Accuracy: 98.92%.	Relies on a limited subset
Ref[22]	Prasanna & Krishna (2024)	ALO, WOA	API Call Sequence dataset	WOA Accuracy: 94.04%, ALO Accuracy: 91.42%.	High computational cost

system. These factors highlight the need for efficient solutions that balance performance with real-time applicability and ease of implementation. This gap has led to the introduction of a new perspective for increasing detection accuracy, reducing the computational burden, and enabling an optimal balance between feature selection and classifier performance.

Additionally, most of the work is dataset-specific and is confined to those specific datasets for a particular model to achieve high performance in different real scenarios. Although metaheuristic optimization algorithms are popular, research has yet to be conducted in the area of feature selection that uses the DMO algorithm for AMD. Table 2. 4 summarizes the metaheuristic optimization algorithms used as feature selection methods for AMD reported in the literature.

3. Methodology

This section reviews the utilized dataset, data preprocessing steps, and feature selection with the use of the modified DMOA algorithm. The modifications to the DMOA Algorithm are also discussed.

3.1 Dataset

This study uses an Android malware dataset known as CICAndMal2017, which was developed by the Canadian Center for Cybersecurity at the University of New Brunswick [23, 24].

Table 2. Feature names in the CICAndMal2017 dataset

#	Feature name	#	Feature name	#	Feature name	#	Feature name
1	Flow ID	22	Flow Packets/s	43	Fwd Packets/s	64	Fwd Avg Bulk Rate
2	Source IP	23	Flow IAT Mean	44	Bwd Packets/s	65	Bwd Avg Bytes/Bulk
3	Source Port	24	Flow IAT Std	45	Min Packet Length	66	Bwd Avg Packets/Bulk
4	Destination IP	25	Flow IAT Max	46	Max Packet Length	67	Bwd Avg Bulk Rate
5	Destination Port	26	Flow IAT Min	47	Packet Length Mean	68	Subflow Fwd Packets
6	Protocol	27	Fwd IAT Total	48	Packet Length Std	69	Subflow Fwd Bytes
7	Timestamp	28	Fwd IAT Mean	49	Packet Length Variance	70	Subflow Bwd Packets
8	Flow Duration	29	Fwd IAT Std	50	FIN Flag Count	71	Subflow Bwd Bytes
9	Total Fwd Packets	30	Fwd IAT Max	51	SYN Flag Count	72	Init_Win_bytes_forward
10	Total Backward Packets	31	Fwd IAT Min	52	RST Flag Count	73	Init_Win_bytes_backward
11	Total Length of Fwd Packets	32	Bwd IAT Total	53	PSH Flag Count	74	act_data_pkt_fwd
12	Total Length of Bwd Packets	33	Bwd IAT Mean	54	ACK Flag Count	75	min_seg_size_forward
13	Fwd Packet Length Max	34	Bwd IAT Std	55	URG Flag Count	76	Active Mean
14	Fwd Packet Length Min	35	Bwd IAT Max	56	CWE Flag Count	77	Active Std
15	Fwd Packet Length Mean	36	Bwd IAT Min	57	ECE Flag Count	78	Active Max
16	Fwd Packet Length Std	37	Fwd PSH Flags	58	Down/Up Ratio	79	Active Min
17	Bwd Packet Length Max	38	Bwd PSH Flags	59	Average Packet Size	80	Idle Mean
18	Bwd Packet Length Min	39	Fwd URG Flags	60	Avg Fwd Segment Size	81	Idle Std
19	Bwd Packet Length Mean	40	Bwd URG Flags	61	Avg Bwd Segment Size	82	Idle Max
20	Bwd Packet Length Std	41	Fwd Header Length	62	Fwd Avg Bytes/Bulk	83	Idle Min
21	Flow Bytes/s	42	Bwd Header Length	63	Fwd Avg Packets/Bulk		

The CICAndMal2017 dataset contains more than 10854 samples (4354 malware samples and 6500 benign samples) that were collected from different sources.

- The malware samples are further classified into four primary families. The four subtypes of malware (Ransomware, Adware, Scareware, and SMSmalware) form 42 different attack types.
- Adware is software that is designed to display unwanted advertisements to increase clicks and views. It comprises 104 applications, including the Ewind, Koodous, Feiwo, Shuanet, Selfmite, and Gooligan families.
- Ransomware is a malicious application that seeks to prevent access to computer resources. It comprises 101 applications for Android devices, such as Charger, Jisut, Koler, and WannaLocker.
- Scareware is malware that compels users to purchase unnecessary and potentially malware applications. It comprises 102

applications, such as AndroidDefender, FakeAV, and FakeApp.

- SMSmalware is an unauthorized call or text message sent to others without the mobile owner's permission (Abuthawabeh & Mahmoud, 2020). It consists of 99 applications, including Ji Fake, Bean Bot, Nandrobox, Fake Mart, Fake Notify, and SMS sniffer families.

The CICAndMal2017 dataset features provide basic information about the behavior of malicious and benign applications and detailed characteristics of the network traffic flow in mobile devices. Such features are crucial for understanding how various malware types (ransomware, adware, scareware, SMS malware, and other malware applications) behave with network protocols and how to distinguish them on the basis of network activity. Table 2. presents the feature list and description of the CICAndMal2017 dataset.

For this study, a subset of 34,000 records with 83 features was selected from the CICAndMal2017 dataset to reduce computational complexity and training time while maintaining a balanced representation of malware and benign samples. To handle class imbalance, SMOTE was applied only to the training data in a stratified manner, generating synthetic samples based on feature similarities to prevent overfitting. A 5-fold stratified cross-validation was used throughout to ensure fair and reliable evaluation, with SMOTE applied exclusively on the training folds and validation data left untouched to avoid data leakage. Final performance metrics—including accuracy, precision, recall, and F1-score—are reported as the average ± standard deviation across the folds, providing a realistic assessment of the model's generalization ability.

3.2 Data preprocessing

Data preprocessing is an essential step in the proposed model. The process includes two main steps: data transformation and data normalization.

a) Data Transformation

Data transformation is an important stage of the preprocessing of ML models—it works as an encoder to encode the given labels as categorical numbers. In the case of malware classification, the label column might contain categorical labels such as benign or malicious labels, which can be transformed into nominal feature vectors of 0 and 1 s for binary classification. This transformation is crucial for data preparation in most ML algorithms, as nearly all of them utilize numerical data.

b) Normalization

Normalization is the process of rescaling values or aligning values to a particular range, often between 0 and 1, and it plays a central role in models that use absolute values for inputs to obtain accurate scaling [13]. In this study, the main goal of the normalization process is to ensure that each feature contributes equally to the model's performance, particularly when the features have different units or scales. In the context of the CICAndMal2017 dataset, normalization was performed via min–max scaling. In particular, the data were centered around a mean of 0 and scaled to a unit variance.

The min–max scaling is computed as Eq. (1):

$$X_{Scaled} = \frac{X - X_{Min}}{X_{Max} - X_{Min}} \quad (1)$$

where X_{Scaled} is the result of the min–max scaling procedure, X is the original value, X_{Min} is the minimum value, and X_{Max} is the maximum value in the column. The min max scaler was utilized for 83 features.

3.3 Oversampling

In CICAndMal2017, the number of samples for each type of malware differs considerably, creating an imbalance in classification process. ML models trained on imbalanced data tend to favor classes with higher sample numbers, but this case might result in biased predictions toward the dominant classes. This bias also leads to artificially high accuracy and misclassifies underrepresented classes, causing the model to be ineffective on the test set, especially for rare malware types.

Biases can be eliminated by oversampling, in which an equal number of samples is created for each type of malware. In this study, samples are duplicated from the underrepresented classes to ensure a balanced dataset while improving the model's effectiveness.

3.4 Feature selection

Feature selection is the process of selecting the most relevant features from a large dataset for use in a machine learning model. The model's performance is enhanced when the features selected provide the most relevant features [25]. The DMOA is effective for feature selection, especially when applied to high-dimensional datasets [12], and is appropriate for AMD. In this study, a modified version of the DMOA was selected for feature selection. The DMOA

1. **Initialize Population**
   - Create n_pop individuals, each represented as a **binary vector** of length n_features.
   - Each vector indicates which features are selected (1) or not (0).
2. **Set Initial Best Solution**
   - Evaluate the **fitness** of the initial population.
   - Store the individual with the **highest fitness score** as the best_solution.
3. **For each iteration (1 to n_iter):**

   Step 1: Alpha Selection
   - Identify the **alpha individual** (the one with the highest fitness) in the population.

   Step 2: Crossover (Mate Selection)
   - **For each individual** in the population:
     1. Randomly **select two parents** from the population.
     2. Perform **crossover** at a random point to create a child.
        - Crossover Point: Random integer between 1 and n_features - 1.

   Step 3: Mutation
   - With probability mutation_rate, **mutate the child** by flipping bits at random feature indices.

   Step 4: Scout Phase
   - With probability scout_rate, introduce a **new random individual** into the population to enhance exploration.

   Step 5: Babysitter Exchange
   - **Evaluate the new population.**
   - If the **best individual** from the new population has a higher fitness score than the best_solution, update it.
   - Remove L * n_pop individuals randomly to maintain diversity.
4. **Evaluate Fitness Metrics**
   - Calculate **fitness scores** for all individuals.
   - Update best_solution if any individual surpasses its current fitness score.
5. **Return Best Solution**

Figure. 1 Pseudocode of modified DMOA

incorporates mutation and crossover functions, thereby improving the exploration of the solution space and increasing the likelihood of finding optimal feature sets.

3.4.1. Proposed modified DMOA

The DMOA is a type of stochastic, population-based metaheuristic algorithm introduced by [26]. This approach draws inspiration from the social and foraging behaviors of dwarf mongooses, also known as Helogale. Although these animals forage in groups, each mongoose meticulously searches for food because feeding is not a collective effort; they exhibit seminomadic behavior, creating sleeping mounds near plentiful food sources and seeking out the next available mound. The DMOA works by adopting a sequence of structured phases that mimic the feeding and social behaviors of these mongooses.

The DMOA is efficient for feature selection and other forms of optimization tasks because of its adaptive search ability. However, as with many optimization techniques, the DMOA faces challenges related to convergence to local optima. In this study, the DMOA's performance is improved by adding adaptive mechanisms, such as crossover operations and mutation strategies. These additions help the system adjust and optimize itself, exploring solutions more thoroughly for better outcomes. These enhancements also help the system diversify the search process, enabling the algorithm to avoid local optima and identify superior feature subsets. Fig.1 outlines the pseudocode of the proposed modified DMOA, and Fig. 2 details the specific modifications introduced. The steps for the modified DMOA are described as follows:

1. Initialize Population: The algorithm starts by randomly generating an initial population of n_pop individuals. Each individual is represented as a binary vector. Each element in this vector corresponds to a feature, where a value of 1 indicates that the feature is selected, and a value of 0 indicates that it is not. This binary representation is crucial for feature selection in the optimization process.

2. Initial Best Solution: The algorithm evaluates each individual's fitness in the initial population before entering the iterative optimization process. The fitness function used in this evaluation is based on two key factors: accuracy and the number of features. The best solution is determined on the basis of the highest fitness score, which is a combination of these factors. This initial best solution serves as a reference point for future comparisons, guiding the optimization process to find the best set of features.

The fitness function (Fit) can be expressed as:

$$Fit = \alpha \times Accuracy - \beta \times \frac{no.\ of\ SF}{Total\ features} \quad (2)$$

where α and β are weighting parameters that control the trade-off between accuracy and subset size. The individual with the highest fitness score is recorded as the initial best solution, serving as a benchmark for subsequent iterations.

3. Iteration Loop (1 to n_iter): This loop runs for a specified number of iterations, allowing the algorithm to improve the population of solutions over time. Each iteration includes several critical steps:

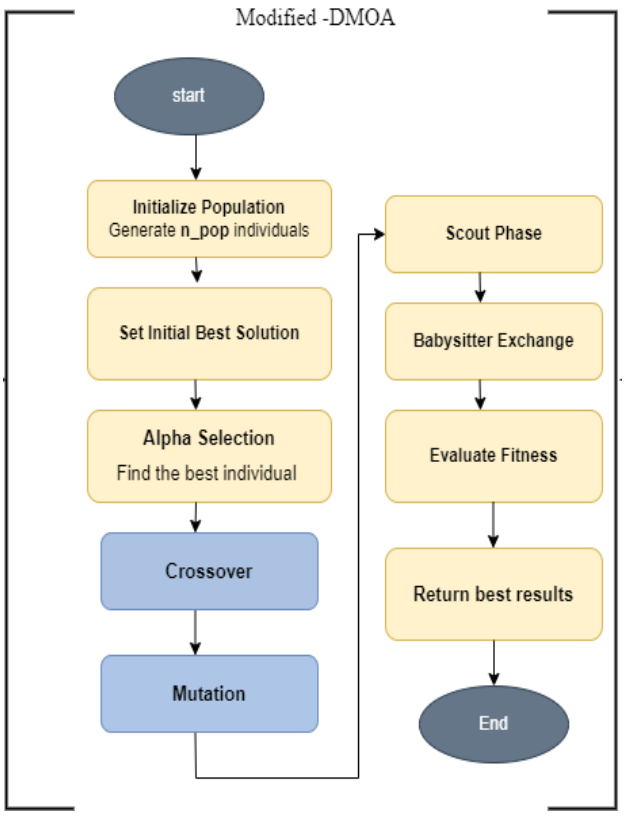

Figure. 2 Block diagram of the modified DMOA

Step 1: Alpha Selection: The algorithm identifies the alpha individual, which is the one with the highest fitness score in the current population. This individual represents the best feature set obtained thus far, influencing the next generation of individuals.

Step 2: Crossover (Mate Selection): For each individual, randomly select two parent individuals from the population and perform a single-point crossover on their binary vectors to generate offspring. The crossover point is chosen randomly between 1 and $n_{features} - 1$.

Step 3: Mutation: In a probability set with a given mutation rate, the child may undergo mutation. In particular, the bits at the feature indices are randomly flipped in the child's binary vector.

Step 4: Scout Phase: With the probability equal to the scout rate, a new random individual is placed into the population. This scout individual can explore the new regions of the solution space and present better solutions that were absent in the current population.

Step 5: Babysitter Exchange: The fitness of the new population, "including any new individuals created from crossover, mutation, or scouts," is evaluated. If the best individual in this new population surpasses the fitness of the previously recorded best solution, then the algorithm updates the best solution. This exchange allows the algorithm to retain the most promising solutions during optimization.

4. Fitness Evaluation: Once all the iterations are completed, the algorithm uses a fitness function to evaluate each individual's fitness in the final population. This function assesses how well the feature set of each individual performs in terms of classification accuracy, providing a final measure of performance.

5. Return to the Best Solution: The algorithm returns the best solution, the individual with the highest fitness score found during the iterations. The final solution represents the optimal feature subset selected using the modified DMOA.

3.5 Classification

The classification stage is a critical component of the AMD model because it determines whether network activity is benign or malicious. When the most relevant features are identified, ML algorithms can train classifiers using labelled data, enabling the model to recognize malware-related patterns. The likelihood of identifying the most effective approach for classifying Android malware was increased by applying several classifiers, including DT, RF, KNN, and XGBoost. These classifiers were selected for their relative strengths when dealing with the CICAndMal2017 datasets.

3.6 Modifications to the DMOA algorithm

The original DMOA was inspired by the social behavior and foraging strategies typical of dwarf mongooses, including their sleeping mounds. However, the changes in the present work involve modifications of the DMOA algorithm to increase its feature selection and optimization efficiency. The main changes in the original DMOA are outlined below.

A) Crossover Function

The crossover operator in the modified DMOA is a major operator that is applied to form new solutions on the basis of the information of two parent solutions. In binary formula-based optimization problems, feature selection crossover keeps the population large and improves the exploitation of the solution space. The crossover function has three stages. In the first stage, a pair of candidate solutions is randomly selected from the population. However, the best two solutions are excluded from the selection process, and genetic diversity is encouraged while avoiding premature convergence. The second stage involves selecting a random crossover point within the binary

representation of the parent solutions. The offspring are created by combining the two parent solutions at this crossover point. This stage allows the creation of a new solution that inherits traits from both parents. In this manner, the likelihood of finding a better solution is increased in subsequent generations.

In the third and final stages, the newly generated offspring are integrated into the population. This outcome is achieved by replacing one of the least fit solutions in the population with offspring. Depending on the algorithm's implementation, the fitness of the offspring might be compared with that of the population. Finally, the best individual solutions are selected for the next generation. The crossover function is applied to search for improved combinations of features; in this study, it is used to improve the results of classification tasks in areas such as malware detection. By merging subsets of the features, the algorithm can select the best result between exploration and exploitation.

B) Mutation

The mutation process is one of the essential parts of the optimization mechanism in the modified DMOA. Mutation occurs after the crossover operation: it creates new offspring from two parents' solutions. The primary aim of mutation is to randomly modify some of the offspring's genetic material, thereby ensuring that the population does not become homogeneous. The mutation occurs with a probability defined by the mutation_rate parameter, which determines how often a child solution undergoes mutation. When a child is selected for mutation, the algorithm randomly selects a set of indices in the child's solution (representing the features selected) and flips their values. This bit-flipping process is the critical operation of mutation, where 0 becomes a 1 and vice versa.

The mutation prevents the algorithm from becoming stuck in suboptimal solutions by introducing small, random changes that might lead to better solutions. When bits in the binary representation of the feature set are flipped, mutation can explore new areas of the solution space that may not have been reached via crossover alone. Subsequently, the algorithm can balance exploration for "searching for new solutions" and exploitation for "refining the best solutions." Through mutation, the algorithm increases its chances of finding the optimal set of features for the classification task, especially in complex problems such as malware detection.

Table 3 summarizes the hyperparameter settings used for the modified DMOA in this study, along with the selection strategies applied. These values were determined based on empirical tuning and preliminary experimentation to ensure convergence and generalization.

Table 3. Hyperparameters of the Modified DMOA

Parmeters	Value	Selection strategy
Number of populations	30	Empirically tuned based on preliminary experiments
Number of Iterations	100	Fixed value ensuring convergence without overfitting
Mutation Rate	0.5	Standard value for binary metaheuristics
Crossover Type	Single point	Enhances genetic diversity during reproduction
Fitness Function	Accuracy-based with subset size penalty	Balances accuracy and dimensionality

4. Implementation and results

This section presents the implementation process and discusses the results of implementing the model via various ML algorithms and compares these results with those of other metaheuristic algorithm models.

4.1 Implementation

The study was conducted in a Windows environment via the VSC IDE, which provides a comprehensive set of tools for developing, testing, and debugging code. The most recent Python version (V3.12) was used for model implementation and evaluation because of its extensive support for ML and the availability of many data analysis packages.

The modified DMOA was applied to select the most relevant features based on a fitness function that considers accuracy and the number of selected features. The fitness function was defined to calculate the error rate (*err*), considering both the prediction accuracy and the number of selected features (*SF*). A lower fitness value (error rate) indicates a better feature subset.

$$err=\alpha * (1 - Accuracy)+ \beta* \left(\frac{No.\ of\ SF}{max\ No.\ of\ F}\right) \quad (3)$$

As a result, 40 features were selected for binary classification from the original 83 features. The selected features were 2, 6, 8, 9, 12, 16, 17, 19, 21, 23, 25, 26, 27, 28, 29, 31, 32, 34, 35, 37, 38, 39, 40,

41, 45, 48, 50, 52, 56, 57, 59, 61, 62, 63, 71, 72, 73, 74, 75, and 79.

The performance of the system was evaluated using metrics such as accuracy, precision, recall, F1 score and convergence time to measure the effectiveness and efficiency of the malware detection approach. The results provide insights into the model's classification, effectiveness, and efficiency. A confusion matrix (CM) was also used to provide a detailed breakdown of the model's predictions across different classes. It offers a thorough analysis of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The mathematical formulations of the evaluation metrics are defined as follows:

$$Accuracy = \frac{(TP + TN)}{N} \tag{4}$$

$$Precision = \frac{(TP)}{(TP + FP)} \tag{5}$$

$$Recall = \frac{(TP)}{(TP + FN)} \tag{6}$$

$$F1\ Score = \frac{2*(Precision * Recall)}{(Precision + Recall)} \tag{7}$$

4.2 Results

The model was evaluated using ML algorithms. The algorithms used are DT, RF, KNN, and XGBoost. The DT classifier achieved perfect performance in binary classification of malware and benign samples, with an accuracy of 1.00, precision of 1.00, recall of 1.00, F1 score of 1.00, and specificity of 1.00. It recorded zero false positives and false negatives, indicating that all samples were correctly classified. The convergence time was 173.59 seconds, demonstrating the model's efficiency in detecting Android malware. Similarly, the XGBoost classifier also achieved perfect performance with an accuracy of 1.00, precision of 1.00, recall of 1.00, and F1 score of 1.00. However, due to the boosting mechanism, the convergence time was significantly higher, taking 425.08 seconds to train and converge. The RF classifier also performed exceptionally well, with an accuracy of 1.00, precision of 1.00, recall of 1.00, and F1 score of 1.00. There were no misclassifications, and the convergence time was 233.72 seconds, suggesting good efficiency. Although some classifiers achieved perfect performance in our cross-validation experiments, we acknowledge that such results may not generalize to other datasets or real-time environments. Fig.3 shows the CM for the DT,RF,and XGBoost classifiers, including the

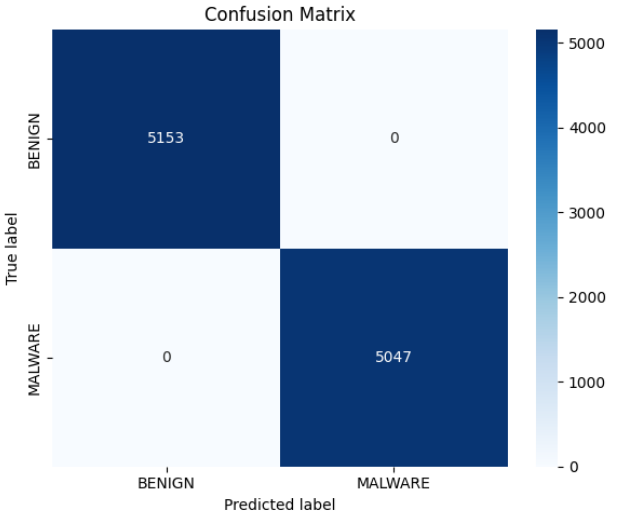

Figure. 3 CM for DT, RF, and XGBoost

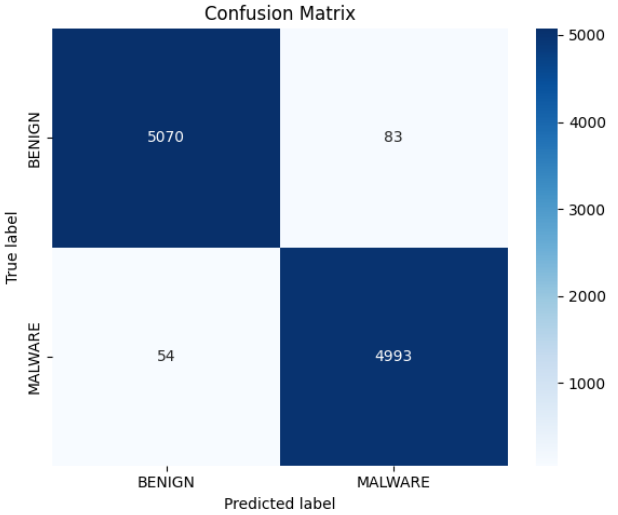

Figure. 4 KNN classifier CM

corresponding values of TP, TN, FP, and FN values for each class.

In contrast, the KNN classifier, while still performing well, showed slightly lower results, with an accuracy of 0.9888, precision of 0.9866, recall of 0.9888, and F1 score of 0.9866. The convergence time for KNN was notably higher, taking 1331.20 seconds due to the computational complexity of distance calculations during prediction. Fig.4 depicts the CM for the KNN classifier and the corresponding TP, TN, FP, and FN values for each class. The binary classification results are summarized in Table 4, which compares the performance of the classifiers.

The evaluation of the proposed modified DMOA, which uses various ML algorithms, has proven the effectiveness of the proposed modified DMOA in enhancing feature selection for AMD. Compared with the original DMOA, all classifiers achieved consistent performance improvements in accuracy.

Table 4. Binary classification results for all classifiers

Classifier	Accuracy	Precision	F1-score	Recall	Time (s)
RF	1.000	1.000	1.000	1.000	233.72
DT	1.000	1.000	1.000	1.000	173.59
XGBoost	1.000	1.000	1.000	1.000	425.08
KNN	0.988	0.986	0.986	0.989	1331.2

Fig.5 shows the accuracy of the ML algorithms when the proposed modified DMOA is used. The proposed modified DMOA algorithm attained improvements in accuracy across the DT, RF, XGBoost and KNN algorithms in malware binary classification. The RF, DT, and XGBoost algorithms had the most significant improvements, achieving perfect accuracy scores of 1.0 when the modified DMOA was applied, in contrast to their original accuracy scores of 0.9992, 0.9367, and 0.9835, respectively, for the original DMOA. These results highlight the significant enhancement in the performance of these algorithms in the classification of malware instances via the modified DMOA. Although the KNN algorithm shows only a modest improvement (from 0.9814 to 0.9888), the findings still indicate the positive impact of the modified DMOA on the algorithm's effectiveness in malware classification.

In terms of precision, the proposed modified DMOA demonstrates constant improvements in precision for the ML algorithms. DT, RF and XGBoost had perfect precision scores of 1.00. The precision of the KNN in the modified DMOA (0.9866) is slightly greater than that of the original DMOA (0.9814), indicating a minor improvement (Fig. 6). The F1 score demonstrates notable improvements across several ML algorithms with the modified DMOA. DT, RF and XGBoost achieved perfect F1 scores of 1.00, increasing from 0.9315 and 0.9820, respectively, in the original DMOA. RF obtained an F1 score of 1.00, compared with 0.9861 in the original DMOA. KNN in the modified DMOA yielded an F1 score of 0.9866, showing a slight improvement over the original DMOA's 0.9814. Fig.7 provides a comparative visualization of the F1 scores between the proposed modified DMOA and the original DMOA for binary classification.

Fig.8 shows a comparison of the convergence times between the proposed modified DMOA and the original DMOA in binary classification. The convergence time reflects the efficiency of the proposed modified DMOA. All the ML algorithms achieved better convergence times. The DT convergence time was reduced to 173.59 from 700.55 in the original DMOA, and the RF's time decreased to 233.72 from 983.83. XGBoost significantly decreased to 425.08 from 1583.82, and KNN decreased to 1331.20 from 1835.59. These findings suggest that the proposed modified DMOA significantly enhances the performance of the classifiers, particularly in terms of classification accuracy and efficiency, thereby demonstrating its effectiveness in Android malware detection using the CICAndMal2017 dataset.

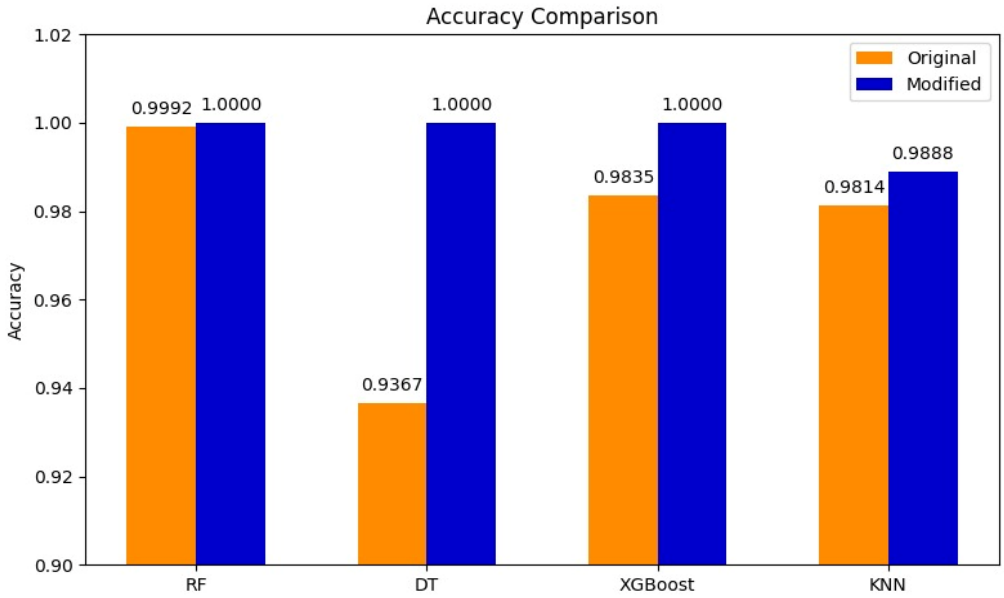

Figure. 5 Accuracy comparison of the proposed modified DMOA and the original DMOA

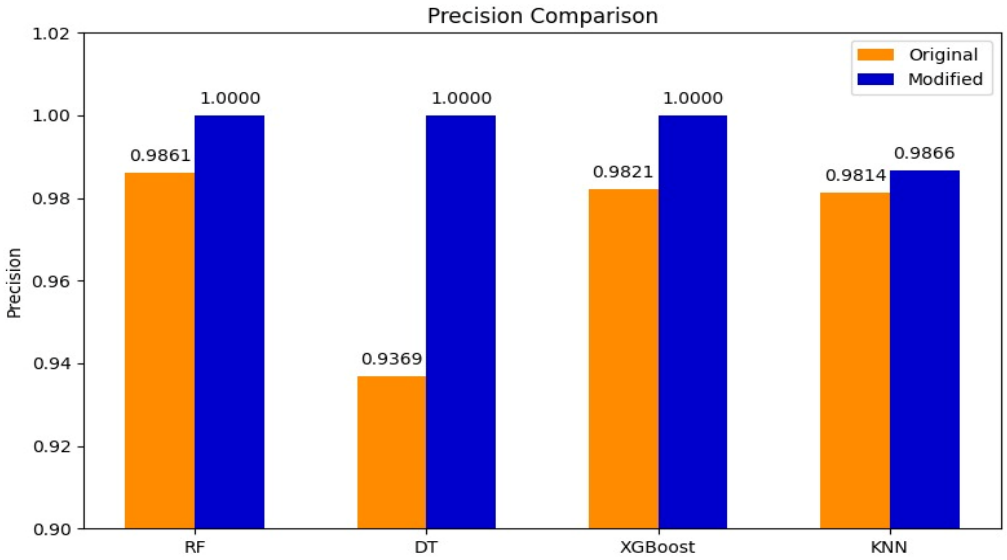

Figure. 6 Precision comparison of the proposed modified DMOA and the original DMOA.

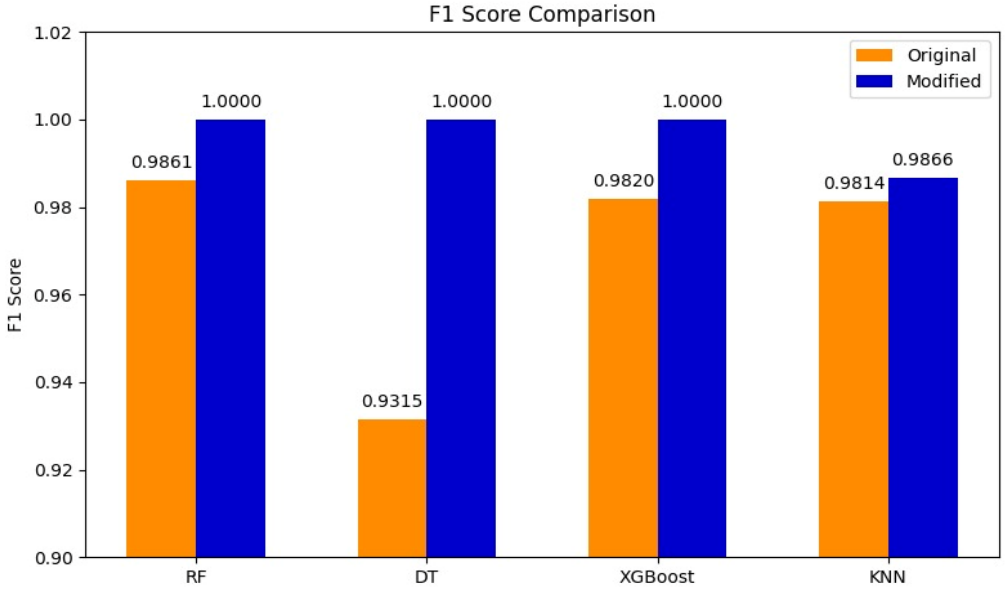

Figure. 7 F1 score comparison of the proposed modified DMOA and the original DMOA

The proposed modified DMOA model was compared with other AMD models. All the following studies utilized the CICAndMal2017 dataset for their models. Table 5. shows a comparison between the proposed modified DMOA results and the other AMD models. A comparative analysis of the proposed model against other AMD models in binary and multiclass classification is also provided. The strengths and weaknesses of each approach in terms of accuracy, precision, F1 score, and convergence time were explored.

The proposed model outperforms other models with a perfect score of 100% in accuracy, precision, and recall for binary classification. A notable comparison is with the GBWO model of Aldehim et al. [16], which achieved an accuracy of 98.59%; the proposed modified DMOA model scored 100%. Thus, the proposed modified DMOA model sets a new benchmark in binary malware classification with its perfect score. Similarly, the proposed model outperforms all the compared models in multiclass classification. RF has an overall accuracy of 100%, surpassing the GBWO model of Aldehim et al. [16], with a score of 98.50%, and the MVO and EMFO models of Taher et al. [14], with a score of 96.9%. Fig.9 presents an accuracy comparison between the

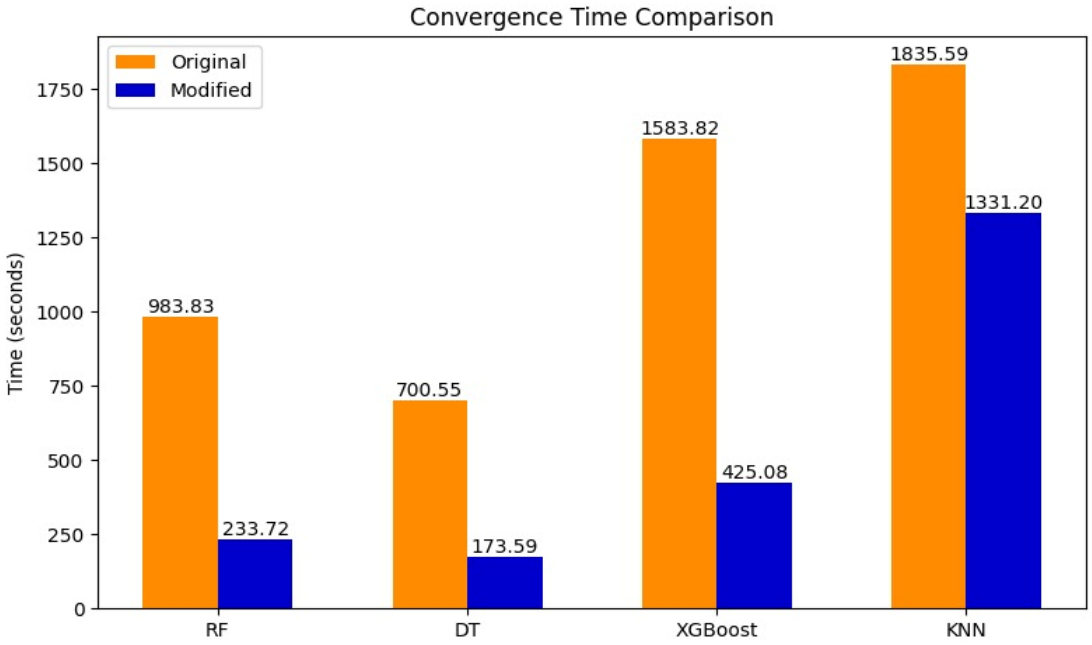

Figure. 8 Convergence time comparison of the proposed modified DMOA and the original DMOA

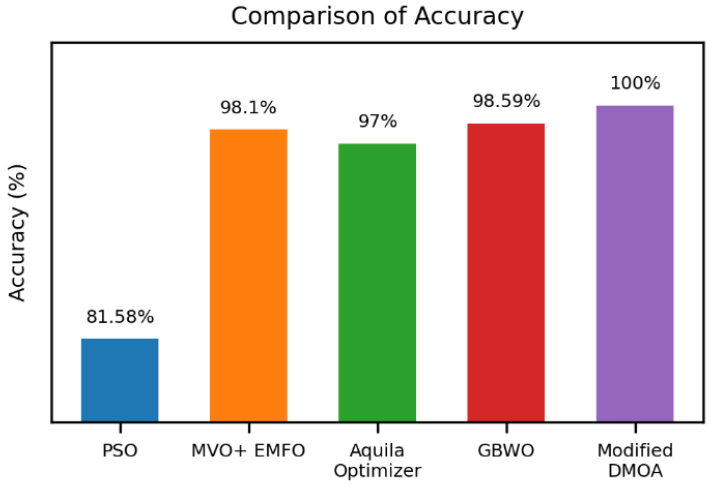

Figure. 9 Accuracy comparison between the proposed modified DMOA and other models

Table 5. Performance comparison of the proposed Modified DMOA model with existing Android malware detection models on the CICAndMal2017 dataset

Ref	Algorithm	Accuracy	Precision	F1 Score
[13]	PSO	81.58%		
[14]	MVO, EMFO	98.1%,	-	-
[15]	Aquila Optimizer	97%	94%	93%
[16]	GBWO	98.59%	-	96.48%
This work	Modified DMOA	100%	100%	100%

proposed modified DMOA and other models in the binary classification of malware. The proposed model outperforms all the other models, achieving a perfect accuracy score of 100% when the DT, RF and XGBoost classifiers are used. The modified DMOA effectively and accurately classified malware, surpassing the performance of other existing models in the task.

5. Conclusion

This study employed the modified DMOA for feature selection and successfully developed an effective ML model for AMD. The CICAndMal2017 dataset was used to evaluate the model's performance.

Several algorithms were applied to binary classification tasks, with the proposed model demonstrating exceptional results—achieving 100% accuracy using DT, RF, and XGBoost classifiers. The modified DMOA consistently improved accuracy, precision, F1 score, recall, and convergence time across all ML algorithms. These improvements are attributed to the method's ability to better identify the most relevant features while minimizing redundant and noisy data, which helps the classifiers perform more efficiently and accurately. The results confirm that the proposed model outperforms existing approaches on the CICAndMal2017 dataset, setting new benchmarks in binary classification and extending the application of DMOA in AMD. Future work could focus on enhancing the performance of the proposed model by combining multiple ML algorithms. However, this integration should be undertaken carefully to prevent reintroducing complexities, ensuring that the model remains straightforward and effective. Testing on distinct Android datasets might include testing the proposed ML model on multiple datasets to prove its applicability in Android systems. Finally, the model's performance could be examined in real-world scenarios, such as integration into existing cybersecurity systems. Finally, a detailed ablation study is planned to quantify the individual impact of the crossover and mutation operations introduced in the modified DMOA, which may support further optimization of the algorithm's performance.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Following are the contributions of the authors: conceptualization, Mosleh Abualhaj; methodology, Rawan Alabdallat and Mosleh Abualhaj; software, Rawan Alabdallat; validation, Rawan Alabdallat and Ahmad Abu-Shareha; formal analysis, Rawan Alabdallat; investigation, Rawan Alabdallat; resources, Mosleh Abualhaj; data curation, Rawan Alabdallat; writing—original draft preparation, Rawan Alabdallat; writing—review and editing, Mosleh Abualhaj and Ahmad Abu-Shareha; visualization, Rawan Alabdallat; supervision, Mosleh Abualhaj; project administration, Mosleh Abualhaj; funding acquisition, Mosleh Abualhaj.

References

[1] Statista, "Annual number of malware attacks worldwide from 2015 to 2023", [Online]. Available: https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/ [Accessed: Dec. 24, 2024].

[2] D. Soi, A. Sanna, D. Maiorca, and G. Giacinto, "Enhancing Android malware detection explainability through function call graph apis", *Journal of Information Security and Applications*, Vol. 80, pp. 103691, 2024.

[3] C. R. Palma, *Malware detection in android applications with machine learning techniques*, 2023.

[4] M. A. H. Saeed, "Malware in computer systems: Problems and solutions", *International Journal on Informatics for Development (IJID)*, Vol. 9, No. 1, pp. 1-8, 2020.

[5] A. Petrosyan, "Malware - Statistics & Facts", [Online]. Available: https://www.statista.com/topics/8338/malware/#topicOverview [Accessed: Dec. 24, 2024].

[6] H. H. R. Manzil and S. M. Naik, "Detection approaches for Android malware: Taxonomy and review analysis", *Expert Systems with Applications*, Vol. 238, pp. 122255, 2024.

[7] N. Z. Gorment, A. Selamat, L. K. Cheng, and O. Krejcar, "Machine learning algorithm for malware detection: Taxonomy, current challenges, and future directions", *IEEE Access*, Vol. 11, pp. 141045-141089, 2023.

[8] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective", *Neurocomputing*, Vol. 300, pp. 70-79, 2018.

[9] O. A. Akinola, J. O. Agushaka, and A. E. Ezugwu, "Binary dwarf mongoose optimizer for solving high-dimensional feature selection problems", *PLOS ONE*, Vol. 17, No. 10, pp. e0274850, 2022.

[10] P. Agrawal, H. F. Abutarboush, T. Ganesh, and A. W. Mohamed, "Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)", *IEEE Access*, Vol. 9, pp. 26766-26791, 2021.

[11] S. M. Abdullah and A. Ahmed, "Hybrid bare bones fireworks algorithm for load flow analysis of islanded microgrids", *Handbook of Research on Fireworks Algorithms and Swarm Intelligence*, pp. 283-314, 2020.

[12] M. A. Elaziz, A. A. Ewees, M. A. Al-Qaness, S. Alshathri, and R. A. Ibrahim, "Feature selection for high dimensional datasets based on quantum-based dwarf mongoose optimization", *Mathematics*, Vol. 10, No. 23, pp. 4565, 2022.

[13] M. S. Hossain, N. Hasan, M. A. Samad, H. M. Shakhawat, J. Karmoker, F. Ahmed, and K.

Choi, "Android ransomware detection from traffic analysis using metaheuristic feature selection", *IEEE Access*, Vol. 10, pp. 128754-128763, 2022.

[14] F. Taher, O. AlFandi, M. Al-Kfairy, H. Al Hamadi, and S. Alrabaee, "DroidDetectMW: a hybrid intelligent model for android malware detection", *Applied Sciences*, Vol. 13, No. 13, pp. 7720, 2023.

[15] M. Grace and M. Sughasiny, "Malware detection for Android application using Aquila optimizer and Hybrid LSTM-SVM classifier", *EAI Endorsed Transactions on Scalable Information Systems*, Vol. 10, No. 1, 2022.

[16] G. Aldehim, M. A. Arasi, M. Khalid, S. S. Aljameel, R. Marzouk, H. Mohsen, and S. S. Ibrahim, "Gauss-mapping black widow optimization with deep extreme learning machine for android malware classification model", *IEEE Access*, Vol. 11, pp. 87062-87070, 2023.

[17] K. Alissa, H. Elkamchouchi, D. Tarmissi, A. Yafoz, R. Alsini, O. Alghushairy, and M. Al Duhayyim, "Dwarf mongoose optimization with machine-learning-driven ransomware detection in internet of things environment", *Applied Sciences*, Vol. 12, No. 19, pp. 9513, 2022.

[18] O. P. Per, "Bat optimization algorithm for wrapper-based feature selection and performance improvement of Android malware detection", *IET Netw.*, Vol. 10, pp. 131-140, 2021.

[19] M. Guendouz and A. Amine, "A new feature selection method based on dragonfly algorithm for android malware detection using machine learning techniques", *International Journal of Information Security and Privacy (IJISP)*, Vol. 17, No. 1, pp. 1-18, 2023.

[20] 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, pp. 102852, 2022.

[21] S. Naick, P. Bethapudi, and S. P. R. Reddy, "Malware detection in android mobile devices by applying swarm intelligence optimization and machine learning for API calls", *International Journal of Intelligent Systems and Applications in Engineering*, Vol. 10, No. 3, pp. 67-74, 2022.

[22] T. L. Prasanna and M. M. Krishna, "Integrating swarm intelligence with machine learning techniques for Android malware detection through API call analysis", *Int. J. Comput. Sci. Trends Technol. (IJCST)*, Vol. 12, No. 2, pp. 1-6, 2024.

[23] University of New Brunswick (UNB), "CICAndMal2017 dataset", [Online]. Available: https://www.unb.ca/cic/datasets/andmal2017.html [Accessed: Oct. 26, 2024].

[24] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification", In: *Proc. of 2018 International Carnahan Conference on Security Technology (ICCST)*, pp. 1-7, 2018.

[25] J. Barrera-García, F. Cisternas-Caneo, B. Crawford, M. Gómez Sánchez, and R. Soto, "Feature selection problem and metaheuristics: a systematic literature review about its formulation, evaluation and applications", *Biomimetics*, Vol. 9, No. 1, pp. 9, 2023.

[26] J. O. Agushaka, A. E. Ezugwu, and L. Abualigah, "Dwarf mongoose optimization algorithm", *Computer Methods in Applied Mechanics and Engineering*, Vol. 391, pp. 114570, 2022.

[27] M. M. Abualhaj, S. Al-Khatib, M. O. Hiari, and Q. Y. Shambour, "Enhancing spam detection using hybrid of Harris Hawks and firefly optimization algorithms", *Journal of Soft Computing and Data Mining (JSCDM)*, Vol. 35, No. 2, pp. 161-174, 2024.

[28] M. M. Abualhaj, M. O. Hiari, A. Alsaaidah, M. Al-Zyoud, and S. Al-Khatib, "Spam feature selection using firefly metaheuristic algorithm", *Journal of Applied Data Sciences*, Vol. 5, No. 4, pp. 1692-1700, 2024.

[29] M. M. Abualhaj, A. A. Abu-Shareha, S. N. Alkhatib, Q. Y. Shambour, and A. M. Alsaaidah, "Detecting spam using Harris Hawks optimizer as a feature selection algorithm", *Bulletin of Electrical Engineering and Informatics (BEEI)*, Vol. 14, No. 3, pp. 111-119, 2025.

[30] Y. Sanjalawe, S. Fraihat, S. Al-E'Mari, M. Abualhaj, S. Makhadmeh, and E. Alzubi, "A review of 6G and AI convergence: Enhancing communication networks with artificial intelligence", *IEEE Open Journal of the Communications Society*, Vol. 6, pp. 2308-2355, 2025.

[31] Y. Sanjalawe, S. Fraihat, M. Abualhaj, S. R. Al-E'Mari, and E. Alzubi, "Hybrid deep learning for human fall detection: A synergistic approach using YOLOv8 and time-space transformers", *IEEE Access*, Vol. 13, pp. 41336-41366, 2025.

[32] Y. Sanjalawe, S. Al-E'mari, S. Fraihat, M. Abualhaj, and E. Alzubi, "A deep learning-

driven multi-layered steganographic approach for enhanced data security", *Scientific Reports*, Vol. 15, No. 1, 2025.

[33] M. M. Abualhaj, Q. Y. Shambour, A. A. Abu-Shareha, S. N. Al-Khatib, and A. Amer, "Enhancing malware detection through self-union feature selection using gray wolf optimizer", *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 37, No. 1, pp. 197-205, 2025.

[34] S. Fraihat, Q. Shambour, M. A. Al-Betar, and S. N. Makhadmeh, "Variational autoencoders-based algorithm for multi-criteria recommendation systems", *Algorithms*, Vol. 17, No. 12, pp. 561, 2024.

[35] Q. Shambour, N. Qandeel, Y. Alrabanah, A. Abumariam, and M. K. Shambour, "Artificial intelligence techniques for early autism detection in toddlers: A comparative analysis", *Journal of Applied Data Sciences*, Vol. 5, No. 4, pp. 1754-1764, 2024.

[36] M. Madi, F. Jarghon, Y. Fazea, O. Almomani, and A. Saaidah, "Comparative analysis of classification techniques for network fault management", *Turkish Journal of Electrical Engineering and Computer Sciences*, Vol. 28, No. 3, pp. 1442-1457, 2020.

[37] A. H. Mohammad, T. Alwada'n, O. Almomani, S. Smadi, and N. ElOmari, "Bio-inspired hybrid feature selection model for intrusion detection", *Computers, Materials and Continua*, Vol. 73, No. 1, pp. 133-150, 2022.

[38] A. Almomani, I. Akour, A. M. Manasrah, O. Almomani, M. Alauthman, E. Abdullah, A. Al Shwait, and R. Al Sharaa, "Ensemble-based approach for efficient intrusion detection in network traffic", *Intelligent Automation & Soft Computing*, Vol. 37, No. 2, 2023.