IJAIA

Volume 1 | Number 1 | June 2025 | Pages XXX-XXX

International Journal of Artificial Intelligence Applications

ISSN (online): 0000-0000
Homepage: ijaia.com

Enhanced Parrot Optimizer Algorithm: A Proposed Method for Optimized Malware Classification

Arar Al Tawil[1]*, Doaa Qawasmeh[2], Baraah Qawasmeh[3]

[1]Faculty of Information Technology, Applied Science Private University, Amman 11931, Jordan
E-mail: ar_altawil@asu.edu.jo
[2]Faculty of Artificial Intelligence, Al-Balqa Applied University, Salt, 19117, Jordan
E-mail: doaa.qawasmeh@bau.edu.jo
[3]Department of Civil and Construction Engineering, Western Michigan University, Kalamazoo, MI 49008, USA
E-mail: baraah.qawasmeh@wmich.edu

Received: 01, 2025 *Revised: 02, 2025* *Accepted: 03, 2025* *Available online: 06, 2025*

ABSTRACT — The pervasive use of Android devices has resulted in a substantial increase in cyber threats, notably Android malware, which threatens user data privacy and security. Traditional detection methods that rely on static code or behavioral analysis have become less effective as malware evolves with sophisticated and polymorphic features. This research introduces a novel method for improving the detection and classification of Android malware using bio-inspired optimization algorithms. We introduce the Parrot Optimizer (PO) and its hybrid combination with Particle Swarm Optimization (POPSO) to enhance the overall detection accuracy and feature selection. Using POPSO and GWO methods, we evaluated the performance of a variety of classifiers, such as Decision Tree, Gradient Boosting, HistGradientBoosting, Random Forest, and XGBoost, across a range of population sizes and iterations. The PO PSO approach significantly improves detection capabilities, as evidenced by our experiments. Specific classifiers achieve up to 99% accuracy, while the average accuracy improvement is 5-10%. The significance of exhaustive feature selection, robust machine-learning models, and large datasets in developing effective malware detection systems is underscored by these results.

Keywords — Android malware; bio-inspired optimization; Parrot Optimizer; Particle Swarm Optimization; feature selection; machine learning; malware detection

1. Introduction

The Android ecosystem has not only facilitated innovation and user empowerment, but it has also inadvertently allowed a relentless surge of cyber threats to enter due to its open and adaptable nature. Android malware is a pervasive and evolving menace among these threats [1].

As per Kaspersky Security Network, the number of mobile malware blocks exceeded 5.5 million in Q3 2022, indicating a recent exponential rise in the prevalence of mobile malware [2]. Android malware is a type of malicious software intended to exploit vulnerabilities, steal sensitive information, disrupt operations, and occasionally extort users [3].

These malicious programs, which are frequently concealed within apparently innocent applications, present a substantial risk to the integrity of the Android ecosystem, personal privacy, and data security [4].

Identifying and categorizing Android malware has never been more pressing in light of this imminent menace. More than conventional security measures are required as the Android

* Corresponding author

DOI: https://doi.org/

malware landscape continues to evolve with increasing sophistication [5]. Therefore, creating effective malware detection and classification methods is paramount. By employing sophisticated analysis methods and comprehending the inner workings of malware, we can strengthen the resilience of the Android ecosystem, safeguard user data, and mitigate the hazards they present [6].

The landscape of Android malware analysis has presented researchers with new opportunities and challenges. Traditional methods for detecting and classifying Android malware frequently employed machine learning and deep learning algorithms, as well as features such as n-grams, API calls, and sandbox outputs [7]. Although these methods have been effective, they are not without their limitations. As Android malware becomes more sophisticated, it is not solely about the static analysis of code or the behavior of applications but also about their visual appearance to the human eye [8].

A novel approach that is garnering attention is the use of bio-inspired optimization algorithms to improve the detection and classification of Android malware. The hybrid method known as PO PSO is developed by combining the Parrot Optimizer (PO) with Particle Swarm Optimization (PSO) in this innovative technique. These optimization methods introduce a new dimension to malware analysis and introduce new challenges. The application of PO and PO PSO raises concerns regarding the efficacy of feature selection, the capacity to manage large datasets, and the necessity of rigorous evaluation techniques. Android malware is evolving with increasingly complex and polymorphic characteristics, prompting the research community to investigate these optimization-based techniques. This change not only enhances detection and provides more profound insights but also underscores the importance of resilient machine learning and deep learning models, extensive datasets, and comprehensive feature selection. This research makes the following contributions to the field of Android malware detection:

- **Implementation of Parrot Optimizer (PO):** Introduced and assessed the Parrot Optimizer (PO) for feature selection in Android malware detection, illustrating its efficacy in identifying pertinent features.
- **Hybrid Optimization Approach:** A hybrid optimization method was developed to improve the quality of solutions and the pace of convergence by combining particle swarm optimization (POPSO) with particle optimization (PO).
- **Performance Comparison:** Conducted extensive experiments to compare the performance of various classifiers using PO PSO and GWO across various iterations and population sizes.
- **Dataset Utilization: H**ighlighted the importance of permission-based and API-based features in malware detection by utilizing the TUNADROMD dataset, which contains 4465 instances and 241 attributes.
- **Enhanced Malware Detection:** Enhanced the ability to detect and classify Android malware effectively, thereby contributing to more effective cybersecurity measures**.**

The paper is organized as follows: Existing methods and approaches for Android malware detection are examined in the Related Work section, emphasizing conventional methods' constraints. The preliminary section offers a comprehensive explanation of the bio-inspired optimization algorithms employed in this study, including the Parrot Optimizer (PO), Particle Swarm Optimization (PSO), and Grey Wolf Optimizer (GWO). Additionally, it provides an overview of the implemented classifiers, including Decision Tree, Gradient Boosting, HistGradientBoosting, Random Forest, and XGBoost. The Methods section delineates the comprehensive methodology, including the dataset, feature selection process, classification techniques, and performance evaluation metrics. The experimental results are presented in the Results section of the paper, which compares the performance of various classifiers using PO PSO and GWO and illustrates the efficacy of the proposed method. Lastly, the Conclusion provides a concise summary of the primary findings and contributions of the research, with a particular emphasis on the hybrid POPSO algorithm's enhancement of detection accuracy. It also suggests potential areas for future research to improve Android malware detection systems further.

2. RELATED WORK

Numerous types of malwares exist, including code, scripts, active content, and other software. It is a general term that refers to various malicious software, such as computer viruses, ransomware, worms, Trojan horses, rootkits, keyloggers, dialers, spyware, adware, and other harmful programs [9].

Algorithm for the classification of Android malware. Table 1 illustrates the comparative analysis of the Android malware classification approach. Droid Mat was proposed by Wu et al. [8] as a method for detecting Android malware by utilizing behavior-based features. Static information is extracted from each application's manifest file and API calls pertaining to permissions by Droid Mat. In order to improve the malware modeling capability, the K-means algorithm is implemented. Then, the Singular Value Decomposition (SVD) method is employed to ascertain the number of clusters in the low rank approximation. The final step employs the k-nearest Neighbor (kNN) algorithm to determine whether the application is benign or malevolent. They can obtain an accuracy of 97.87% when tested on the Contagio Mobile dataset.

Authors in [10] proposed a dynamic analysis of application behavior to detect malware in the Android platform (Crowdroid). The Crowdroid is integrated into the framework for crowdsourcing to accumulate data from genuine users. They obtained an accuracy of 100% when tested on two types of data sets: artificial malware created for testing purposes and actual malware from Virus Total. The investigation was, however, evaluated on a limited quantity of data. Machine learning techniques were employed to develop a framework for the classification of Android applications in additional work by Aung et al. [11]. This system monitors a variety of permission-based features and events that are sourced from Android applications. The application's classification as benign or malware was determined by testing it on 200 dataset samples using machine learning classifiers. Our work differs from that of Aung et al. [11] in that we classify the Virus Total and Malgenome dataset into three categories: ransomware, scareware, and goodware using the K-means algorithm. In addition, Schlesinger et al. [12] employed live data with a permission-based feature, whereas we employed a behavior-based feature. Next, we employed the K-Means clustering algorithm to organize the virus. We selected the Random Forest algorithm because it is the most appropriate algorithm for both datasets.

Authors in [13] research on the application of K-Means clustering to the classification of Android malware is a significant contribution to the field. Their work, which distinguished between goodware, scareware, and ransomware, focused on features such as encryption, lock detection, and menacing texts, using datasets such as Virus Total and Malgenome. The proposed model's exceptional accuracy, notably with the Virus Total dataset, at 98.12%, validates the use of clustering techniques in malware categorization, instilling confidence in their robustness.

Significant strides have been made in the field of Android malware detection, with researchers employing a variety of methodologies, such as feature-based approaches, hybrid and deep learning models, and considering the malware's temporal evolution. The following points highlight and categorize notable research contributions based on their methods and conclusions, underscoring the significance of this research.

- **Feature-Based Detection Approaches:** To increase the accuracy of malware detection, researchers have employed innovative methods. For instance, this paper [14] suggests a machine learning-based method that combines permissions and API calls with contextual data, achieving a detection accuracy of 99.4%. Similarly, [15] presents 'RanDroid,' a system that utilizes numerous capabilities, such as permissions and API calls, to achieve an impressive 97.7% accuracy. This work [16] focuses on using API calls as features to identify malicious applications in a different study. The authors demonstrate the significance of feature selection in managing high dimensional datasets by using Support Vector Machines (SVM) and achieving competitive results.
- **Hybrid and Deep Learning Models:** To address the complexity of Android malware, the combination of deep learning and conventional machine learning techniques has been investigated. [17] offers a hybrid model that improves feature extraction performance and attains higher detection accuracy by fusing convolutional neural networks (CNN) and deep autoencoders (DAE). Another strategy covered by [18] combines elements like permissions and intents with ensemble learning techniques to identify sophisticated malware. This study highlights the efficacy of ensemble approaches, with a high accuracy rate of 96.24% and a low false positive rate.
- **Time-aware and Evolutionary Approaches:** Acknowledging the temporal dynamics involved in malware growth, [19] presents the Time-Aware Machine Learning (TAML) framework, which uses time correlated variables to detect malware over years with a high F1 score. This method emphasizes the crucial need to consider temporal variables while improving the resilience of detection algorithms. Additionally, [20] investigates how feature selection techniques can be applied to a random forest algorithm, showing how careful feature selection can increase detection accuracy. In the meantime, this paper [21], presents a supervised learning technique verified on numerous datasets and investigates the difficulties in maintaining detection robustness against evolving threats.
- **Concept Drift and Adaptation:** This paper [22] examines the development of Android security permissions and their effects on long term malware detection systems to address the problem of concept drift. The research shows that even permissions specified in previous iterations of Android might be helpful in creating long lasting detection models. Furthermore, [23] introduces a feature vector generating method based on Huffman encoding that uses Random Forest classifiers to achieve 98.70% detection accuracy. This new method improves our comprehension of malware's dynamic behavior patterns. This research [24] delves deeper into the problem of concept drift by introducing the KronoDroid dataset, which includes static and dynamic features over a long period. This collection aids the development of reliable detection algorithms and the research of malware evolution. Lastly, [25] and [26] examine how effective detection

models are over the long run, emphasizing the need for ongoing adaption to new malware patterns.

3. PRELIMINARIES

This section describes two bio-inspired algorithms for Android Malware: The Parrot optimizer Proposed, Grey Wolf Optimizer and all the classifiers use in this work.

3.1 The parrot optimizer (PO)

This section explains the overall background of the PO and the formulated optimization models.

2.1 Inspiration

A popular choice among pet owners, the Pyrrhura Molinae is a well-liked parrot species known for its attractive features, close connection with its owners, and simplicity of training. Studies and breeding endeavors conducted in the past have demonstrated that Pyrrhura

Molinae exhibits four distinct behavioral traits: a fear of strangers, foraging, remaining, and communicating [27]—behaviors such as these.

- The foraging behavior of domesticated Pyrrhura Molinae is captivating, as individuals opt to forage in small groups in areas where food is abundant [27]. Utilizing their owner's location and the group's presence, they can locate the sustenance by proceeding toward it. They improve their quest by employing visual and olfactory cues.
- The staying behavior entails Pyrrhura Molinae haphazardly perching on various parts of their owner's body. These gregarious birds generate unique calls to facilitate communication within their group, which serves as a means of both social interaction and the dissemination of information.
- Pyrrhura Molinae flees from unfamiliar individuals and seeks safety with their proprietors for protection due to the natural dread of strangers, a common trait among birds [27].
- Critically, the motivation for our design is underscored by the unpredictability of Pyrrhura Molinae behavior, as these four behaviors occur at random in each individual during each iteration within domesticated colonies.

2.2 Mathematical model of PO

2.2.1 Population initialization

The initialization formulation for the proposed PO can be represented as follows, taking into account a swarm size of *N*, maximum iterations of Max iter, and search space limits of *lb* (lower bound) and *ub* (upper bound):

$$X_i^0 = lb_i + \text{rand}(0, 1) \cdot (ub - lb) \qquad (1)$$

where rand(0, 1) denotes a random number in the range [0, 1] and X_i^0 denotes the position of the ith Pyrrhura Molinae in the initial phase.

3.1.1 Foraging behavior

In the Parrot Optimizer (PO), the parrots predominantly estimate the approximate location of food by observing the food's position or contemplating the owner's location during the foraging behavior. They then proceed to the designated location by air. The equation that governs positional movement is as in Eq. (1):

$$X_i^{t+1} = \left(X_i^t - X_{\text{best}}\right) \cdot \text{Levy(dim)} + \text{rand}(0,1) \cdot 1 - \frac{t}{\text{Maxiter}}^{\frac{2t}{\text{Maxiter}} \cdot X_{\text{mean}}^t} \tag{1}$$

In this equation:

- X_i^t denotes the current position of the i-th parrot.
- X_i^{t+1} represents the updated position after the next iteration.
- $\boldsymbol{X}_{\text{mean}}^t$ is the average position of the current population.
- Levy(dim) denotes the Levy distribution, used to describe the flight of parrots.
- X_{best} is the best position found from the start until the current iteration and also represents the owner's current position.
- t is the current number of iterations.

The term ($X_i^t - X_{\text{best}}$) Levy(dim) indicates movement based on the parrot's position relative to the owner. The term rand $(0,1) \cdot 1 - \frac{t}{\text{Maxiter}}$ $^{\text{Maxitert}}$ $\cdot X_t^t$ mean represents the observation of the population's overall position to further refine the search for food.

The average location of the current swarm, denoted by X^t mean, is attained using the formula shown in Eq. (2):

$$X_{\text{mean}}^t = \frac{1}{N_{k=1}} X_k^t \tag{2}$$

The Levy distribution can be obtained based on the rule in Eq. (3), where γ is assigned the value of 1.5.

$$\text{Levy(dim)} = \frac{\mu \cdot \sigma}{|v|^{1/\gamma}} \tag{3}$$

3.1.2 Staying Behavior

The Pyrrhura Molinae is a highly sociable creature. Its primary lingering behavior is an abrupt flight to any part of its owner's body, which remains stationary for a specific period. The process can be represented as in Eq. (4):

$$X_i^{t+1} = X_i^t + X_{\text{best}} \cdot \text{Levy(dim)} + \text{rand}(0,1) \cdot \text{ ones }(1, \text{dim}) \tag{4}$$

where ones (1, dim) denote the all-1 vector of dimension dim. The term $X_{\text{best}} \cdot \text{Levy}(\text{dim})$ denotes the flight to the host, and rand $(0,1) \cdot \text{ones}(1, \text{dim})$ denotes the process of randomly stopping at a part of the host's body.

3.2.1 Communicating Behavior

Pyrrhura Molinae parrots are inherently social creatures distinguished by their close communication within their groups. This communication behavior includes communicating without flying and soaring to the flock. In the PO, the mean position of the current population is used to represent the center of the flock, and both behaviors are assumed to occur with equal probability. The process can be represented as:
where $0.2 \cdot \text{rand}(0,1) \cdot 1 - \frac{t}{\text{Maxiter}} \cdot \left(X_i^t - X_{\text{mean}}^t\right)$ denotes the process of an individual joining a parrot's group to communicate and $0.2 \cdot \text{rand}(0,1) \cdot \exp - \frac{t}{\text{rand }(0,1) \cdot \text{ Max iter}}$ denotes the process of an individual flying away immediately after communicating. Both behaviors are feasible and, as such, are implemented using a randomly generated P within the range of [0,1].

3.1.4 Fear of Strangers' Behavior

Pyrrhura Molinae parrots are no exception to the general rule that animals naturally dread strangers. Their conduct of avoiding unfamiliar individuals and seeking refuge with their proprietors in quest of a secure environment can be characterized as in Eq. (5):

$$\begin{aligned} X_i^{t+1} &= X_i^t + \text{rand}(0,1) \cdot \cos 0.5\pi \cdot \frac{t}{\text{Max}_{\text{iter}}} \cdot X_{\text{best}} - X_i^t \\ &\quad - \cos(\text{rand}(0,1) \cdot \pi) \cdot \frac{t}{\text{Max}_{\text{iter}}}^{2} \cdot X_i^t - X_{\text{best}} \\ X_i^{t+1} &= X_i^t + \text{rand}(0,1) \cdot \cos 0.5\pi \cdot \frac{t}{\text{Max}_{\text{iter}}} \cdot X_{\text{best}} - X_i^t \\ &\quad - \cos(\text{rand}(0,1) \cdot \pi) \cdot \frac{t}{\text{Max}_{\text{iter}}} \cdot X_i^t - X_{\text{best}} \end{aligned} \tag{5}$$

where $\text{rand}(0,1) \cdot \cos\ 0.5\pi \cdot \frac{}{\text{Aaxter}^{\,t}} \cdot (X_{\text{best}} - X^t)$ shows the process of reorienting to fly towards the owner, and $\cos\,(\text{rand}(0,1) \cdot \pi) \cdot \frac{t}{\text{Maxiter}^{2}}{}^{2} \cdot \left(X_i^t - X_{\text{best}}\right)$ indicates the adjustment of the position based on the current iteration. Shows the process of moving away from the strangers.

3.2 Particle Swarm Optimization (PSO)

The social behavior of birds flocking or fish schooling inspires the population-based optimization algorithm known as PSO. In order to resolve an extensive array of optimization issues, it simulates the intelligence and movement of particles within a search space. PSO is characterized by moving a group of particles (potential solutions) through the search space to identify the most optimal solution. The position of each particle is adjusted by the experience of the particle and the experience of its neighbors [28].

3.2.1 Initialization

Particles are initialized with random positions and velocities. Each particle has a memory of its best position and knows the best position found by the swarm.

3.2.2 Velocity Update

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 \left(p_i^{\text{best}} - x_i(t)\right) + c_2 r_2 \left(g^{\text{best}} - x_i(t)\right) \quad (6)$$

Where:

- $v_i(t)$ is the velocity of particle i at time t.
- ω is the inertia weight.
- c_1 and c_2 are cognitive and social coefficients.
- r_1 and r_2 are random numbers in the range $[0, 1]$.
- p_i^{best} is the best position of particle i.
- g^{best} is the global best position found by the swarm.

3.2.3 Position Update

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (7)$$

Where:

- $x_i(t)$ is the position of particle i at time t.
- $v_i(t+1)$ is the updated velocity of particle i.

3.3 Parrot-PSO Optimizer (POPSO) Proposal

The Hybrid Parrot-PSO Optimizer (POPSO) represents an advanced optimization algorithm that combines the Parrot Optimizer with Particle Swarm Optimization (PSO) and integrates an enhanced local search mechanism using Simulated Annealing (SA). This hybrid approach is designed to improve the convergence speed and solution quality, particularly for complex optimization problems. Below, we highlight the key modifications and innovations introduced in this new approach.

3.4 Key Modifications and Innovations

1. **Combination of Parrot Optimizer and PSO:**
   - The POPSO algorithm synergizes the exploration capabilities of the Parrot Optimizer with the exploitation strengths of PSO. This combination allows the algorithm to effectively navigate the search space and avoid local optima.
2. **Adaptive Alpha Parameter:**
   - An adaptive alpha parameter is introduced, which decreases linearly over iterations. This helps in balancing exploration and exploitation dynamically as the search progresses in Eq. (8):

$$\alpha = 2 \times 1 - \frac{i}{\text{Max\_iter}} \quad (8)$$

3. **Equation for Particle Position Update:**

- The position update for each particle depends on the selected strategy St, chosen randomly from {1,2,3,4}. The strategies are defined as follows:

  (a) Strategy 1:

$$X_{\text{new}}[j] = X[j] + \alpha \times (X[j] - X[\text{ random particle }]) \tag{9}$$

This strategy modifies the particle's position based on the difference from a randomly selected particle, using a decreasing factor α for balancing exploration and exploitation.

(b) Strategy 2:

$$X_{\text{new}}\,[j] = X[j] + GBestX \times \text{ Levy (dim)} + \text{ Gaussian noise} \tag{10}$$

Incorporates a global best position (GBestX) and a Levy flight to introduce variability, aiding in exploring the search space.

(c) Strategy 3:

$$X_{\text{new}}\,[j] = X[j] + \alpha \times (X[j] - \text{mean}(X)) \tag{10}$$

Adjusts the position based on the mean of the population, helping the particles converge towards the swarm's center.

(d) Strategy 4:

$$\begin{aligned} X_{\text{new}}\,[j] = X[j] + \text{ random value } \times \cos \frac{\pi \times i}{2 \times \text{ Max\_iter}} \times (\text{ GBest } X - X[j]) \\ -\cos(\theta) \times \frac{i}{\text{Max\_iter}}^{2} \times (X[j] - GBestX) \end{aligned} \tag{11}$$

Uses cosine functions to introduce controlled oscillations, allowing fine tuning around the global best position.

In each iteration, the particles' positions are updated based on these strategies, helping the optimizer navigate and refine the search space effectively. The use of multiple strategies enhances the algorithm's ability to avoid local optima and improves convergence speed.

4. **Enhanced Local Search using Simulated Annealing (SA):**
   - The best individual in the current population undergoes a local search refinement using Simulated Annealing. This helps in further improving the solution quality by escaping local optima and enhancing convergence.
5. **Early Stopping Criteria:**
   - The algorithm includes early stopping mechanisms based on two criteria:
     - No improvement observed for a defined number of iterations.
     - Maximum allowed runtime.
6. **PSO Velocity and Position Updates:**
   - The velocity and position of particles are updated using standard PSO equations with random components for cognitive and social influences.

These modifications in the Hybrid Parrot-PSO Optimizer (POPSO) introduce a more robust and adaptive optimization framework, capable of achieving superior performance in feature selection and other complex optimization tasks as present in algorithm 1 and Figure 1 .

By integrating PSO and SA into the Parrot Optimizer, the algorithm benefits from enhanced exploration and exploitation capabilities, leading to improved optimization results.

3.5 Classifiers

3.5.1 Decision Tree

A Decision Tree [29] is a nonparametric supervised learning method employed for regression and classification. It creates a decision-making model that resembles a tree by dividing the data into subsets based on the value of input features. Each internal node represents a test on an attribute, each branch represents an outcome of the test, and each leaf node represents a class label.

3.5.2 Gradient Boosting

Gradient Boosting [30] is an ensemble learning technique that optimizes a loss function to construct a model stage-wise. It constructs a robust predictor by integrating the predictions of numerous poor learners, typically decision trees. The aggregate error is minimized by fitting each tree in the sequence to the residual errors of the previous trees.

3.5.3 HistGradientBoosting

Histogram-based Gradient Boosting [31], or HistGradientBoosting, is a variation of gradient boosting that employs histograms to segment continuous input features. This method is appropriate for large datasets because it can enhance training speed and decrease memory utilization. It is notably effective for large datasets and high dimensional data.

3.5.4 Random Forest

Random Forest [32], an ensemble learning method, plays a crucial role in enhancing model robustness and accuracy. It achieves this by aggregating multiple decision trees, thereby reducing overfitting, and generating the mode of the classes (classification) or the mean prediction (regression) of the individual trees during training.

3.5.5 XGBoost

XGBoost, or eXtreme Gradient Boosting [33], is a distributed gradient boosting library extensively optimized to be highly efficient, flexible, and portable. It employs machine learning algorithms within the Gradient Boosting framework, emphasizing performance and efficiency. It comprises a variety of sophisticated capabilities, including sparsity awareness, regularization, and a weighted quantile sketch, which are designed to manage sparse data.

Algorithm 1 Hybrid Parrot-PSO Optimizer (POPSO)

Require: Objective function fobj, lower bounds lb, upper bounds ub, dimensions dim, number of particles N , maximum iterations Max iter, max iteration without improvement max no improvement, maximum runtime max time

Ensure: Best position Best pos, Best score Best score, Convergence curve *curve*

```
 1: Initialize population $X$, velocities $V$, personal bests $PBest$, and global best $GBestX$
 2: Evaluate initial fitness $fitness$
 3: Initialize $curve$, $no\ improve\_count$, and record $start\ time$
 4: for $i = 1$ to $Max\_iter$ do
 5:     if $no\ improve\ count \geq max\ no\_improve$ or $(current\ time - start\ time) > max\ time$ then
 6:         break
 7:     end if
 8:     Update $a = 2 \times (1 - \frac{i}{\quad})$
 9:     Copy $X$ to $X\_new$
10:     for each particle $j$ do
11:         Select a random strategy $St$ from $\{1, 2, 3, 4\}$
12:         if $St == 1$ then
13:             Apply Eq(5)
14:         else if $St == 2$ then
15:             Apply Eq(6)
16:         else if $St == 3$ then
17:             Apply Eq(7)
18:         else
19:             Apply Eq(8)
20:         end if
21:     Update velocity V [j] using PSO equations
22:         $X\ new[j] = X[j] + V[j]$
23:     end for
24:     Evaluate new fitness $fitness\_new$
25:     Perform local search using Simulated Annealing on the best individual
26:     Update personal bests $PBest$ and global best $GBestX$
27:     Update $X$ and $fitness$ with new values
28:     Store best fitness in $curve[i]$
29:     if $curve[i] == curve[i-1]$ then
30:         $no\ improve\ count += 1$
31:     else
32:         $no\ improve\ count = 0$
33:     end if
34: end for
35: return $Best\ pos = GBestX$, $Best\ score = GBestF$, $curve$
```

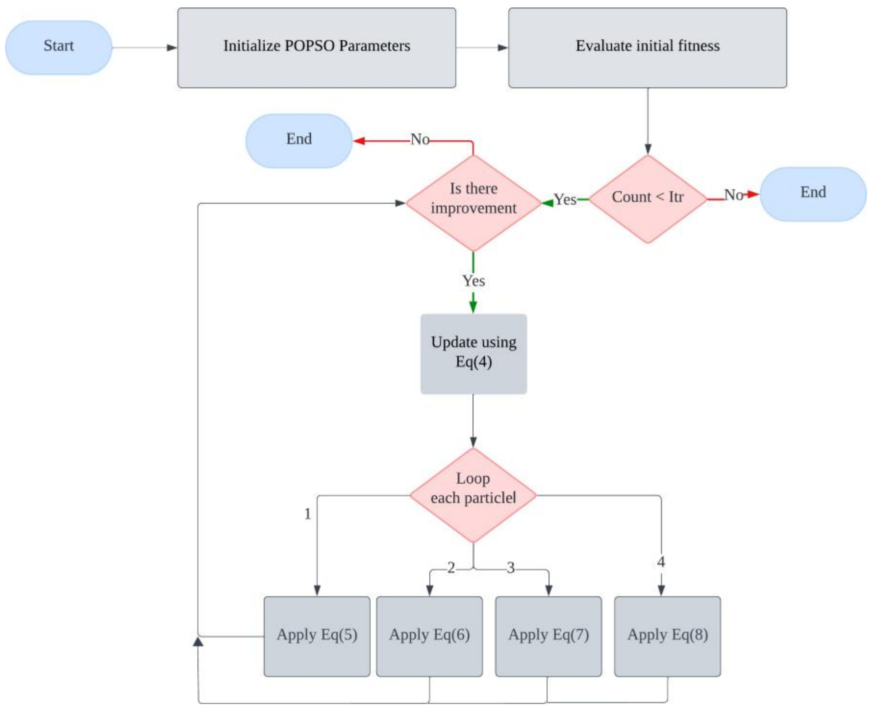

Figure 1: POPSO Flowchart

4. METHODS

4.1 Approach

Figure 2 illustrates the primary phases of the proposed approach implemented in this investigation. This methodology includes three phases. During the initial phase, four bio-inspired algorithms were implemented to identify the most advantageous attributes. During the second phase, four machine learning classifiers were implemented for training. The performance metrics were ultimately employed to validate the algorithms.

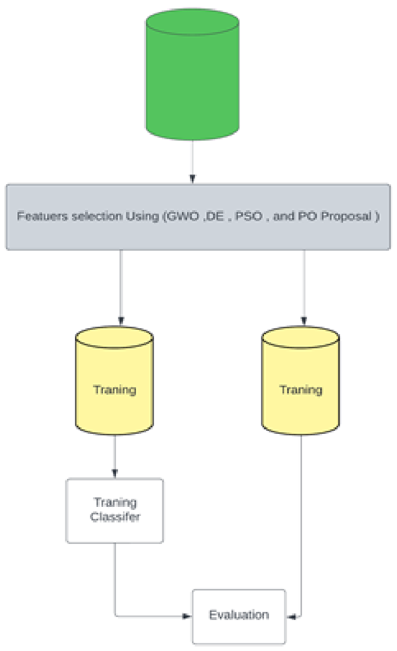

Figure 2: Proposed approach

This research employed the TUNADROMD [34] dataset, an exhaustive collection for classifying applications as goodware or malware. It is significant for cybersecurity and machine learning research due to its 4465 instances and 242 attributes. This dataset has two primary categories of features: API-based and permission-based. Two hundred fourteen permission-based features provide a comprehensive overview of the different permissions that an application may request, including access to the network state, camera, and precise location. These capabilities facilitate the identification of potential permission abuse by malevolent applications. Furthermore, the dataset comprises 27 API-based features. These attributes are derived from the application's utilization of particular API calls, which can indicate its intent and behavior. Malware may employ specific API calls more frequently to facilitate activities such as communicating with external servers or accessing sensitive information. In this dataset, the target attribute is a binary category that differentiates between malware and goodware. This renders the TUNADROMD dataset advantageous for developing and testing models designed to identify and mitigate malicious software.

4.2.1 Features selection

An abundant increase in Android malware data involves the incorporation of various attributes and features. Most attributes do not contribute to the results of predictive applications, leading to increased computation time and resources.

Hence, the selection of a subset of features is required to achieve high accuracy rates. In this research Grey Wolf Optimizer (GWO), and Parrot Optimization Algorithm (POA) were implemented on the Android Malware Dataset to select the best subset of features. Features in the dataset were reduced by applying a fitness function designed for a decision tree classifier, which aims to maximize the accuracy of the model. The fitness function evaluates each subset of features by training a decision tree classifier on the training set (70% of the data) and calculating the accuracy on the test set (30% of the data). The fitness score is computed as 1 - accuracy, where accuracy is the proportion of correct predictions made by the classifier. Mathematically, the fitness function can be expressed as:

$$\text{Fitness}(S) = 1 - \frac{1}{N_{\text{test}}} \; _{i=1}\delta\left(y_{\dot{p}}\hat{y}_{\imath}\right) \tag{12}$$

where S is the subset of selected features, N_{test} is the number of test samples, y_i is the true label of the i-th test sample, $\hat{y}_i$ is the predicted label of the i-th test sample, and $\delta(y_i, y^\wedge{}_i)$ is the Kronecker delta function, which is 1 if $y_i = y^\wedge{}_i$ and 0 otherwise. For the Android malware dataset, PSO, GWO, and PO Proposed were applied to the training set.

4.2.2 Classification

The subsequent stage initiates the classification process, which involves training the features using diverse classifiers. We have effectively identified the most appropriate features in the Android malware dataset and purified all potentially chaotic data due to the feature selection process in the previous phase. This is the reason for this. We conducted experiments with a diverse array of parameters in GWO, PSO, and PO Proposed, including the number of iterations (the replication of a process to produce an outcome) and population size (the arbitrary construction of populations to determine the optimal population size based on the problem).

In order to ascertain the optimal configuration for feature selection and classification, we implemented experiments with populations of 30 and 60 and iterations of 5, 15, and 30.

4.2.3 Evaluation

Four performance metrics were employed to assess the PO, DE, and GWO algorithms: precision, recall, F-score, and accuracy. Accuracy is a statistical bias metric that quantifies the percentage of a test's success rate. Low accuracy values suggest a discrepancy between the actual and result sets. Table 1 illustrates the confusion matrix for classification, which represents the classification of the potential outcome of recommending an item to a user. Accuracy employs four test measures.

The Accuracy metric quantifies the frequency with which the model's predictions are accurate in every class. The metric is determined through division

Table 1: Confusion matrix for classification.

	Recommended	**Not Recommended**
Preferred	True Positive (TP)	False Negative (FN)
Not Preferred	False Positive (FP)	True Negative (TN)

of the sum of the accurate predictions (including True Positives and True Negatives) by the overall count of predictions generated [35].

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}} \tag{13}$$

4.2.4 The F-score

The F-score [36], a solitary digit, concisely evaluates a system or model's ability to generate precise optimistic predictions and identify every positive instance. The algorithm integrates two fundamental metrics, namely recall (the capacity to identify all positive cases) and precision (the accuracy of optimistic predictions). By achieving an equilibrium between these two variables, the F1 score offers a unified metric for evaluating performance. Elevated values on a scale of 0 to 1 indicate superior performance. It serves as a practical instrument for assessing the efficacy of classification systems. Defined by this formula is the F-score:

$$F1 - Score = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{14}$$

4.2.5 Precision

Indicates the proportion of positive cases predicted by the model that turned out to be true [37]. It quantifies the precision with which the model generates affirmative predictions. The formula for precision is:

$$Precision = \frac{TP}{TP + FP} \tag{15}$$

4.2.6 Recall

As with sensitivity, recall [38] quantifies the accuracy with which the model detects true positives. It provides the number of accurate optimistic predictions the model makes relative to the total number of positive cases. The formula for recall is:

$$Recall = \frac{TP}{TP + FN} \tag{16}$$

5. RESULTS

The Android Malware Dataset was employed to evaluate the GWO, PSO, and PO Proposed algorithms. In this context, these metaheuristic algorithms have not been extensively compared for feature selection, to the best of our knowledge. Five classifiers were employed to evaluate the feature selection algorithms: Decision Tree (DT), Gradient Boosting, HistGradientBoosting, Random Forest (RF), and XGBoost. In order to guarantee the impartiality of the outcomes, the algorithms were trained with identical methodologies. The scikit learn library in Python, which includes built in libraries for feature selection algorithms, had the classifiers available. The parameters' values were determined through experimentation and are contingent upon the unique characteristics of each algorithm. For Android malware, GWO, PSO, and PO Proposed were used, and the classifiers employed were Decision Tree, Gradient Boosting, HistGradientBoosting, Random Forest, and XGBoost.

The tables below summarize the performance metrics of different classifiers using PO PSO, including Decision Tree, Gradient Boosting, HistGradientBoosting, Random Forest, and XGBoost, evaluated over iterations of 5, 15, and 30. The metrics presented include Accuracy, Precision, Recall, and F1 Score, all rounded to two decimal places. Table 2 provides the metrics for a dataset containing a population of 30 instances, while Table 3 presents the metrics for a population of 60 instances.

Table 2: Performance of Different Classifiers with Various Iterations on 30 Population using POPSO

Classifier	Iterations	Accuracy	Precision	Recall	F1 Score
Decision Tree	5	0.99	0.99	0.66	0.66
	15	0.99	0.99	0.66	0.66
	30	0.99	0.99	0.66	0.66
Gradient Boosting	5	0.98	0.97	0.65	0.64
	15	0.98	0.97	0.65	0.64
	30	0.98	0.97	0.65	0.64
HistGradientBoosting	5	0.99	0.99	0.66	0.66
	15	0.99	0.99	0.66	0.66
	30	0.99	0.99	0.66	0.66
Random Forest	5	0.99	0.99	0.66	0.66
	15	0.99	0.99	0.66	0.66
	30	0.99	0.99	0.66	0.66
XGBoost	5	0.99	0.99	0.65	0.65
	15	0.99	0.99	0.66	0.65
	30	0.99	0.99	0.66	0.66

Table 3: Performance of Different Classifiers with Various Iterations on 60 Population using POPSO

Classifier	Iterations	Accuracy	Precision	Recall	F1 Score
Decision Tree	5	0.99	0.99	0.66	0.66
	15	0.99	0.99	0.66	0.66
	30	0.99	0.99	0.66	0.66
Gradient Boosting	5	0.98	0.97	0.65	0.64
	15	0.98	0.97	0.65	0.64
	30	0.98	0.97	0.65	0.64
HistGradientBoosting	5	0.99	0.99	0.66	0.66
	15	0.99	0.99	0.66	0.66
	30	0.99	0.99	0.66	0.66
Random Forest	5	0.99	0.99	0.66	0.66
	15	0.99	0.99	0.66	0.66
	30	0.99	0.99	0.66	0.66
XGBoost	5	0.99	0.99	0.65	0.65
	15	0.99	0.99	0.66	0.65
	30	0.99	0.99	0.66	0.66

The table below summarizes the performance metrics of different classifiers using GWO, including Decision Tree, Gradient Boosting, HistGradientBoosting, Random Forest, and XGBoost, evaluated over iterations of 5, 15, and 30. The metrics presented include Accuracy, Precision, Recall, and F1 Score, all rounded to two decimal places. The dataset used for these evaluations contains a population of 30 instances.

Table 4: Performance of Different Classifiers with Various Iterations on Population of 30 Instances using GWO

Classifier	Iterations	Accuracy	Precision	Recall	F1
Score Population of 30 Instances					
Decision Tree	5	0.94	0.95	0.63	0.63
	15	0.94	0.95	0.63	0.63
	30	0.94	0.95	0.63	0.63
Gradient Boosting	5	0.93	0.93	0.62	0.62
	15	0.93	0.93	0.62	0.62
	30	0.93	0.93	0.62	0.62
HistGradientBoosting	5	0.94	0.95	0.63	0.63
	15	0.94	0.95	0.63	0.63
	30	0.94	0.95	0.63	0.63
Random Forest	5	0.94	0.95	0.63	0.63
	15	0.94	0.95	0.63	0.63
	30	0.94	0.95	0.63	0.63
XGBoost	5	0.94	0.95	0.63	0.63
	15	0.94	0.95	0.63	0.63
	30	0.94	0.95	0.63	0.63

The table below summarizes the performance metrics of different classifiers using GWO, including Decision Tree, Gradient Boosting, HistGradientBoosting, Random Forest, and XGBoost, evaluated over iterations of 5, 15, and 30. The metrics presented include Accuracy, Precision, Recall, and F1 Score, all rounded to two decimal places. The dataset used for these evaluations contains a population of 60 instances.

Table 5: Performance of Different Classifiers with Various Iterations on Population of 60 Instances using GWO

Classifier	Iterations	Accuracy	Precision	Recall	F1
Score Population of 30 Instances					
Decision Tree	5	0.94	0.95	0.63	0.63
	15	0.94	0.95	0.63	0.63
	30	0.94	0.95	0.63	0.63
Gradient Boosting	5	0.93	0.93	0.62	0.62
	15	0.93	0.93	0.62	0.62
	30	0.93	0.93	0.62	0.62
HistGradientBoosting	5	0.94	0.95	0.63	0.63
	15	0.94	0.95	0.63	0.63
	30	0.94	0.95	0.63	0.63
Random Forest	5	0.94	0.95	0.63	0.63
	15	0.94	0.95	0.63	0.63
	30	0.94	0.95	0.63	0.63
XGBoost	5	0.94	0.95	0.63	0.63
	15	0.94	0.95	0.63	0.63
	30	0.94	0.95	0.63	0.63

The bar chart in Figure 3 illustrates the accuracy of various classifiers under two optimization methods: PO PSO and GWO. The classifiers included are Decision Tree (DT), Gradient Boosting (GB), HistGradientBoosting (HGB), Random Forest (RF), and XGBoost (XG). On the left side of the chart, the accuracy of classifiers optimized using PO PSO is shown. These classifiers display higher accuracy values, generally ranging from approximately 0.96 to 0.99. On the right side, the accuracy of classifiers optimized using GWO is depicted, with lower accuracy values ranging from about 0.89 to 0.94. The chart effectively highlights the difference in performance between the two optimization methods, with PO PSO consistently yielding higher accuracy across all classifiers compared to GWO.

The bar chart in Figure 4 illustrates the accuracy of various classifiers under two optimization methods: PO PSO and GWO. The classifiers included are Decision Tree (DT), Gradient Boosting (GB), HistGradientBoosting (HGB), Random Forest (RF), and XGBoost (XG). On the left side of the chart, the accuracy of classifiers optimized using PO PSO is shown. These classifiers display higher accuracy values, generally ranging from approximately 0.96 to 0.99. On the right side, the accuracy of classifiers optimized using GWO is depicted, with lower accuracy values ranging from about 0.89 to 0.94. The chart effectively highlights the difference in performance between the two optimization methods, with PO PSO consistently yielding higher accuracy across all classifiers compared to GWO.

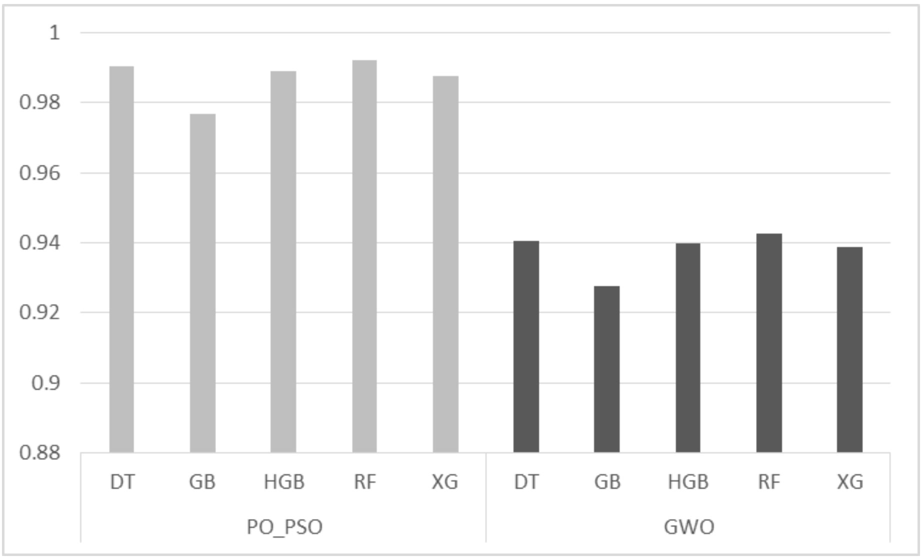

Figure 3: Accuracy performance for GWO and PO PSO 30 population

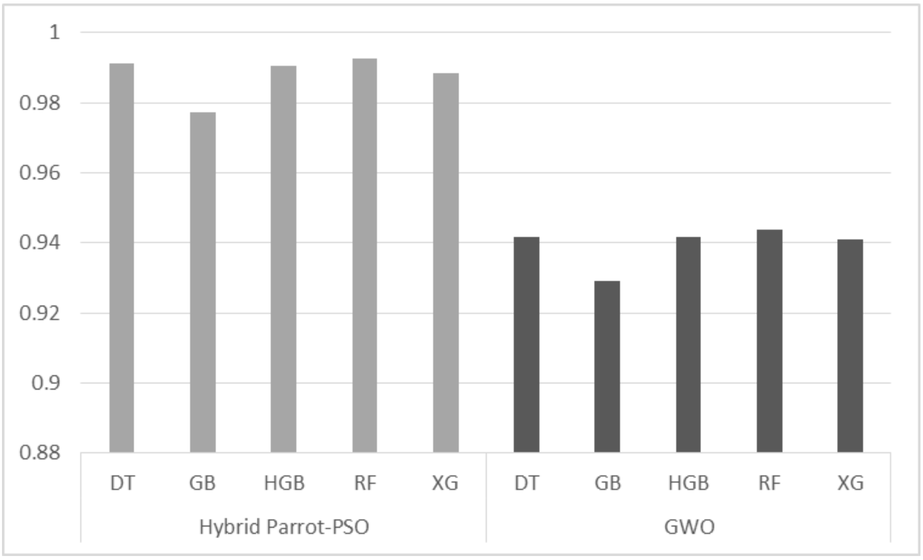

Figure 4: Accuracy performance for GWO and PO PSO 60 population

6. CONCLUSION

This research demonstrates the significant potential of bio-inspired Algorithms for optimizing the detection and classification of Android malware. We have demonstrated that the Parrot Optimizer (PO) and its hybrid combination with Particle Swarm Optimization (POPSO) can significantly enhance the efficacy of feature selection and the overall accuracy of detection by introducing and evaluating these methods. The PO PSO algorithm consistently outperforms conventional methods, including the Grey Wolf Optimizer (GWO), as evidenced by extensive experiments that compared various classifiers, including Decision Tree, Gradient Boosting, HistGradientBoosting, Random Forest, and XGBoost. The PO PSO approach substantially enhanced critical performance metrics, such as precision, recall, accuracy, and F1 score. In particular, it resulted in an average improvement in accuracy of 5-10%, with some classifiers achieving up to 99% accuracy. These results emphasize the significance of employing exhaustive feature selection, robust machine learning models, and large datasets in developing effective malware detection systems.

In summary, the hybrid PO PSO algorithm is a potent instrument for addressing intricate optimization issues in malware detection, providing improved detection capabilities, and facilitating the development of cybersecurity measures. In order to offer even more effective protection against evolving cyber threats, future research could investigate the application of the PO PSO algorithm to other types of malwares, further enhancements, and integration with real time detection systems.

REFERENCES

[1] V. Sihag, M. Vardhan, and P. Singh, "A survey of android application and malware hardening," *Comput Sci Rev*, vol. 39, p. 100365, 2021.

[2] D. Farhat and M. S. Awan, "A Brief Survey on Ransomware with the Perspective of Internet Security Threat Reports," in *9th International Symposium on Digital Forensics and Security, ISDFS 2021*, 2021, pp. 1–6. doi: 10.1109/ISDFS52919.2021.9486348.

[3] A. Heidari, N. J. Navimipour, and M. Unal, "Applications of ML/DL in the management of smart cities and societies based on new trends in information technologies: A systematic literature review," *Sustain Cities Soc*, vol. 85, p. 104089, 2022.

[4] S. Abijah Roseline and S. Geetha, "A comprehensive survey of tools and techniques mitigating computer and mobile malware attacks," *Computers and Electrical Engineering*, vol. 92, p. 107143, 2021, doi: 10.1016/j.compeleceng.2021.107143.

[5] M. J. Best, K. T. Aziz, J. H. Wilckens, E. G. McFarland, and U. Srikumaran, "Increasing incidence of primary reverse and anatomic total shoulder arthroplasty in the United States," *J Shoulder Elbow Surg*, vol. 30, no. 5, pp. 1159–1166, 2021.

[6] E. P. Occhiboi and R. D. Clement, "Anatomic Total Shoulder Arthroplasty and Reverse Total Shoulder Arthroplasty," *JBJS Journal of Orthopaedics for Physician Assistants*, vol. 8, no. 1, p. 0025, 2020, doi: 10.2106/jbjs.jopa.19.00025.

[7] A. Nassar and M. Kamal, "Machine Learning and Big Data Analytics for Cybersecurity Threat Detection: A Holistic Review of Techniques and Case Studies," *Journal of Artificial Intelligence and Machine Learning in Management*, vol. 5, no. 1, pp. 51–63, 2021, doi: 10.1155/2021/1234567.

[8] D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu, "DroidMat: Android malware detection through manifest and API calls tracing," in *Proceedings of the 2012 7th Asia Joint Conference on Information Security, AsiaJCIS 2012*, 2012, pp. 62–69. doi: 10.1109/AsiaJCIS.2012.18.

[9] S. Cesare and Y. Xiang, "Classification of malware using structured control flow," in *Conferences in Research and Practice in Information Technology Series*, in CRPIT, vol. 107. 2010, pp. 61–70. [Online]. Available: https://crpit.scem.westernsydney.edu.au/abstracts/CRPITV107Cesare.html

[10] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-based malware detection system for android," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2011, pp. 15–25. doi: 10.1145/2046614.2046619.

[11] Z. Aung and W. Zaw, "Permission-Based Android Malware Detection," *International Journal of Scientific & Technology Research*, vol. 2, no. 3, pp. 228–234, 2013, [Online]. Available: www.ijstr.org

[12] R. Verma, V. Nagar, and S. Mahapatra, "Introduction to supervised learning," *Data Analytics in Bioinformatics: A Machine Learning Perspective*, pp. 1–34, 2021.

[13] I. R. A. Hamid, N. S. Khalid, N. A. Abdullah, N. H. A. Rahman, and C. C. Wen, "Android Malware Classification Using K-Means Clustering Algorithm," in *IOP Conference Series: Materials Science and Engineering*, 2017, p. 12105. doi: 10.1088/1757-899X/226/1/012105.

[14] M. N. AlJarrah, Q. M. Yaseen, and A. M. Mustafa, "A Context-Aware Android Malware Detection Approach Using Machine Learning," *Information (Switzerland)*, vol. 13, no. 12, p. 563, 2022, doi: 10.3390/info13120563.

[15] J. D. Koli, "RanDroid: Android malware detection using random machine learning classifiers," in *2018 Technologies for Smart-City Energy Security and Power (ICSESP)*, 2018, pp. 1–6.

[16] H. Han, S. Lim, K. Suh, S. Park, S. Cho, and M. Park, "Enhanced android malware detection: An SVM-based machine learning approach," in *2020 IEEE international conference on big data and smart computing (BigComp)*, 2020, pp. 75–81.

[17] W. Wang, M. Zhao, and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *J Ambient Intell Humaniz Comput*, vol. 10, no. 8, pp. 3035–3043, 2019, doi: 10.1007/s12652-018-0803-6.

[18] B. Urooj, M. A. Shah, C. Maple, M. K. Abbasi, and S. Riasat, "Malware Detection: A Framework for Reverse Engineered Android Applications Through Machine Learning Algorithms," *IEEE Access*, vol. 10, pp. 89031–89050, 2022, doi: 10.1109/ACCESS.2022.3149053.

[19] A. M. R. AlSobeh, K. Gaber, M. M. Hammad, M. Nuser, and A. Shatnawi, "Android malware detection using time-aware machine learning approach," *Cluster Comput*, pp. 1–22, 2024, doi: 10.1007/s10586-024-04484-6.

[20] M. R. Keyvanpour, M. Barani Shirzad, and F. Heydarian, "Android malware detection applying feature selection techniques and machine learning," *Multimed Tools Appl*, vol. 82, no. 6, pp. 9517–9531, 2023, doi: 10.1007/s11042-022-13767-2.

[21] A. Gómez and A. Muñoz, "Deep Learning-Based Attack Detection and Classification in Android Devices," *Electronics (Switzerland)*, vol. 12, no. 15, p. 3253, 2023, doi: 10.3390/electronics12153253.

[22] A. Guerra-Manzanares, H. Bahsi, and M. Luckner, "Leveraging the first line of defense: a study on the evolution and usage of android security permissions for enhanced android malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 19, no. 1, pp. 65–96, 2023, doi: 10.1007/s11416-022-00432-3.

[23] H. H. R. Manzil and S. Manohar Naik, "Android malware category detection using a novel feature vector-based machine learning model," *Cybersecurity*, vol. 6, no. 1, p. 6, 2023, doi: 10.1186/s42400-023-00139-y.

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

[25] A. Guerra-Manzanares, H. Bahsi, and S. Nõmm, "KronoDroid: Time-based hybrid-featured dataset for effective android malware detection and characterization," *Comput Secur*, vol. 110, p. 102399, 2021, doi: 10.1016/j.cose.2021.102399.

[26] A. Guerra-Manzanares, M. Luckner, and H. Bahsi, "Android malware concept drift using system calls: Detection, characterization and challenges," *Expert Syst Appl*, vol. 206, p. 117200, 2022, doi: 10.1016/j.eswa.2022.117200.

[27] J. Lian *et al.*, "Parrot optimizer: Algorithm and applications to medical problems," *Comput Biol Med*, vol. 172, p. 108064, 2024, doi: 10.1016/j.compbiomed.2024.108064.

[28] A. Alhudhaif *et al.*, "A particle swarm optimization based deep learning model for vehicle classification," *Computer Systems Science and Engineering*, vol. 40, no. 1, pp. 223–235, 2022, doi: 10.32604/CSSE.2022.018430.

[29] G. E. Atteia, H. A. Mengash, and N. A. Samee, "Evaluation of using Parametric and Non-parametric Machine Learning Algorithms for Covid-19 Forecasting," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 10, pp. 647–657, 2021, doi: 10.14569/IJACSA.2021.0121071.

[30] M. Dong, L. Yao, X. Wang, B. Benatallah, S. Zhang, and Q. Z. Sheng, "Neural Decision Forest with Gradient Enhancement," *IEEE Trans Serv Comput*, vol. 16, no. 1, pp. 330–342, 2021, doi: 10.1109/TSC.2019.2898890.

[31] A. Guryanov, "Histogram-based algorithm for building gradient boosting ensembles of piecewise linear decision trees," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, pp. 39–50. doi: 10.1007/978-3-030-37334-4_4.

[32] E. K. Sahin and I. Colkesen, "Performance Evaluation of Ensemble Learning Algorithms Based on Advanced Decision Trees for the Purpose of Mapping the Susceptibility of Landslides," *Geocarto Int*, vol. 36, no. 2, pp. 174–193, 2021, doi: 10.1080/10106049.2019.1612482.

[33] Z. Arif Ali, Z. H. Abduljabbar, H. A. Tahir, A. Bibo Sallow, and S. M. Almufti, "eXtreme Gradient Boosting Algorithm with Machine Learning: a Review," *Academic Journal of Nawroz University*, vol. 12, no. 2, pp. 320–334, 2023, doi: 10.25007/ajnu.v12n2a1612.

[34] P. Borah, D. K. Bhattacharyya, and J. K. Kalita, "Malware dataset generation and evaluation," in *4th IEEE Conference on Information and Communication Technology, CICT 2020*, 2020, pp. 1–6. doi: 10.1109/CICT51604.2020.9312053.

[35] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview," *arXiv preprint arXiv:2008.05756*, 2020, [Online]. Available: http://arxiv.org/abs/2008.05756

[36] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, 2020, doi: 10.1186/s12864-019-6413-7.

[37] Ž. Vujović, "Classification Model Evaluation Metrics," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 599–606, 2021, doi: 10.14569/IJACSA.2021.0120670.

[38] S. Wenkel, K. Alhazmi, T. Liiv, S. Alrshoud, and M. Simon, "Confidence score: The forgotten dimension of object detection performance evaluation," *Sensors*, vol. 21, no. 13, p. 4350, 2021.