sciendo

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **24**, No 2
Sofia • 2024 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2024-0020

African Vulture Optimization-Based Decision Tree (AVO-DT): An Innovative Method for Malware Identification and Evaluation through the Application of Meta-Heuristic Optimization Algorithm

Praveen Kumar Kaithal, Varsha Sharma

School of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India
E-mails: praveenkaithal@yahoo.com varshasharma@rgpv.ac.in

Abstract: *Malware remains a big threat to cyber security, calling for machine learning-based malware detection. Malware variations exhibit common behavioral patterns indicative of their source and intended use to enhance the existing framework's usefulness. Here we present a novel model, i.e., African Vulture Optimization-based Decision Tree* (*AVO-DT*) *to increase the overall optimization.*

The datasets from Android apps and malware software train the AVO-DT model. After training, the datasets are pre-processed by removing training errors. The DT algorithm is used by the developed AVO model to carry out the detection procedure and predict malware activity. To detect malware activities and improve accuracy, such an AVO-DT model technique employs both static and dynamic methodologies. The other measurements on Android applications might be either malicious or benign. Here we also developed malware prevention and detection systems to address ambiguous search spaces in multidimensionality difficulties and resolve optimization challenges.

Keywords**:** *Android, Malware detection, Malicious software, Android malware detection, Machine learning, Classification.*

1. Introduction

Intelligent smartphone gadgets and complex sensors have revolutionized the realm of associated and ease in the rapidly evolving digital technology. Software designed to compromise computer operations, obtain confidential data, or breach protected computer networks is referred to as malware. Programs that inadvertently cause harm owing to flaws are not considered malware as their definition is based on their malevolent intent and their violation of what the user expects. Sometimes, both deliberately destructive software and actual malware are referred to as "badware". These pose a threat to the internet's accessibility, the reliability of its recipients, and the confidentiality of its users since they are designed to obtain access to computer systems and network assets, interfere with computer processes, and collect data about individuals without the creator of the system's agreement. Malware propagation has

impacted many aspects of daily life, including social networks [2], digital automation [3], e-Governance [1], and mobile networks [4]. Software infection can take many different forms, including viruses, worms, trojan horses, rootkits, backdoors, botnets, spyware, and adware. Because the aforementioned malware types do not conflict with one another, a single malware sample may exhibit traits from multiple categories simultaneously.

Typically, detection models should be periodically (e.g., regularly) revamped based on the collected data from outdoors to be conscious of malware evolution. Regardless, this leads to damaging attacks, namely auxiliary pathway attacks that compromise the expanding expertise and create escape routes for regulated malware experiments. As of right moment, we are aware of no prior research that looked into this fundamental problem with Android malware locators [5]. This type of malware has been a common threat that targets mobile devices in recent times.

The authors of malware employ obfuscation techniques [6] such as guidance replacement, register reassignment, procedure restructuring, dead code deployment, code transposition, and code integration to avoid being detected by firewalls, antivirus software, and gateways to other networks These security measures usually rely on signature-based methods, which make it difficult to identify fraudulent activity software programs that have not been before observed. Zero-day infections cannot be immediately protected against by commercial antivirus providers because these must first be analyzed for them to produce indications. Static or dynamic malware evaluation approaches are frequently employed to get beyond the limitations of signature-based methodologies. The methods for analyzing malware assist investigators in comprehending the dangers and motivations connected to a harmful code snippet. With this enhanced information, one can respond to current developments in malware generation or take proactive steps to mitigate possible future attacks. Unidentified malware can be grouped and categorized into pre-existing families using characteristics that are obtained from the examination of malware. Several scholars have examined the problem and offered theories and methods from different perspectives. This outdated strategy is readily circumvented by contemporary viruses, consequently, novel strategies built around structure and behaviors are required. In such circumstances, machine learning can be useful. Furthermore, there are times when the method is not enough to manage a large enough dataset [15]. A variety of techniques, such as ML with a checking scheme [17], MLDroid [16], and others, were developed to tackle this issue, but no practical solution has yet been found. By selecting certain features, one can train a classifier to identify important behavioral characteristics in an almost limitless supply of malware. The optimization models and methodologies utilized in economic crises, natural disasters, wars, and pandemics are reviewed here by the author [32]. Furthermore, an effort has been taken to highlight opportunities for model formulation and optimization efforts that have not yet been investigated, as well as shortcomings in the study of one type of event.

The existing research papers explore various opportunities for detecting malware and developing intelligent response systems. To detect malware, K a k a v a n d, M o h a m m a d and A l i [8] used SVM and KNN, and training was

provided based on the flagged data points. Neither a dynamic approach nor hybrid malware detection strategies are included in their models. Lopes et al. [9] examination of several Machine Learning (ML) algorithms focused on mobile malware and examined methods based on permissions, API calls, used features, permissions, and calls, and those based on both. It struggles with data imbalance and small data sets. A deep learning framework Droid Deep Learner was proposed by Wang et al. [12]. To get API function calls and permissions, their approach analyses Java source code as well as manifest files, enabling users to access all the functionality of Android apps.

This novel proposed paper presents the following contributions:

- **Design an Algorithm for detecting and analyzing the malware.** This paper provides the meta-heuristic algorithm optimal selection built on AVO-DT Algorithm for Android malware detection and analysis, here they obtain a novel model i.e., African Vulture Optimization-based Decision Tree (AVO-DT) to increase the overall optimization for detecting and analyzing the malware. There has been a numerous optimization algorithm in the area of malware detection specific to machine learning, android, and a few surveys on static and dynamic analysis.
- **Guide for malware analysts.** Finally, it is realized that the contribution claimed in this paper will help, guide, and assist researchers and malware analysts in getting appropriate using such proposed model presented model successfully predicts whether the provided application is malicious or benign and was created for malware detection and protection for their domain-specific analysis.
- Creating an innovative rotational search formula for optimization tasks.
- Offering a strong and effective algorithm with a short execution time, little difficulty in computation, and good performance.

The research work is structured as follows: Section 2 provides some background information on Android applications, such as the Android system designs, security features, and categories for Android malware. Section 3 provides a thorough analysis of machine learning-based methods for detecting malware, data collection from different sources, data pre-processing, feature selection, machine learning approaches, computations, and detection efficiency assessment. Section 4 makes some recommendations for potential study topics and obstacles to be overcome. Finally, Section 5 presents our conclusions.

2. Literature review

Deep learning models aim to enhance prediction accuracy through a metaheuristic approach, that learning model more deeply in this context. To prove the SEDMDroid's viability, Zhu et al. [7] present testing results on two different datasets collected using the static assessment approach. The initial analysis focuses on features like permission, sensitive programming interfaces, monitoring frameworks, etc. that are frequently used in Android malware, and SEDMDroid achieves 89.07% accuracy in terms of these asymmetrical unchanging parts. The next dataset accurately distinguishes sensitive information stream data as highlights with

an average precision of 94.92%. Positive test outcomes demonstrate that the suggested approach is a powerful means of identifying Android malware.

According to A l z u b a i d i [10], the proliferation of cell phones across the globe has spurred the development of millions of applications that are both free and profitable. The aforementioned apps enable users to carry out many tasks, such as gaming, communication, and completing financial and educational tasks. These commonly used devices have been increasingly identified by dangerous malicious programming since they routinely store sensitive personal information. The concepts and risks associated with malware are the main focus of this article, which also examines the systems and approaches currently in use to differentiate malware in terms of methodology, relevant datasets, and evaluation metrics.

Based on a Synthesis Proportion (CR) of consent matches, K a t o, S a s a k i and S a s a s e [11] suggest where Android malware is located. A consent pair's percentage of all responses in an application's database is how we define the CR. Our main focus was on how unnecessary credentials lead to a generally low CR in malware. We build knowledge bases on the CR to obtain insights without using frequency information. The ability to compare ratings is determined for every software application based on the available data sets. Eight scores are finally processed into elements for AI (ML) based classification algorithms. It is possible to achieve steady exposition in this way. Our features consist of only eight layers; thus the suggested plot requires a shorter period to prepare and can work with other machine learning-based plans. Furthermore, our main points can statistically provide unambiguous data that helps people comprehend geographical results. Since we can meet all of the needs, our approach makes sense for practical usage. Our results show that our strategy can accurately detect malware with up to 97.3% accuracy using real datasets. Furthermore, in comparison with an existing plan, our plan can reduce the element aspects by about 100% while maintaining equivalent precision on recent datasets.

Focusing on the subsequent development attack against Android malware IDs was motivated by L i et al. [13]. Without revealing the preparation information, an additional advancement is constructed and subtly incorporated into the model. It is then activated upon the introduction of an application containing the trigger that initiates the event. We demonstrate the suggested attack on four common malware investigators that have received a lot of attention from academics. Our analysis demonstrates that the suggested secondary passage attack successfully avoids more than 750 malware tests with an evasion rate of up to nearly 100%. Furthermore, the aforementioned successful attack is supported by a small number of products (only four components) and a very low understanding damaging rate (0.3%).

G o n g et al. [14] An improved and reasonable solution to these shortcomings would be to enable early detection of overlay-based malware in the course of the utilization market investigation procedure, to preserve all of the capabilities of overlays. Because of this, we first conduct an extensive look into the combined features of both benign and malicious applications in the present research. We then implement the Overlay Checker structure to automatically identify overlay-based malware for one of the largest Android application marketplaces in the world. To be

more precise, we have made conscious efforts in the areas of designing, UI research, copying engineering, and run-time environment. As a result, we have been able to maintain outstanding recognition preciseness (97% accuracy and 97% review) and a short per application investigation time (1.7 min) with only two ware servers, despite having an increased workload of 10,000 recently uploaded tasks every day.

Another method for finding malware known as ransom that uses a transformative AI approach is presented by A l m o m a n i et al. [18]. The synthesized minority oversampling purposes method (wiped out) for implementation is employed in conjunction with the help Vector Machines (SVM) calculation to adjust the hyperparameters of the configuration calculation and determine highlights. The data that was utilized dataset, which includes 10,153 Android applications – 500 of which have been classified as ransomware – was compiled from many different places. The proposed method Destroyed tBPSO-SVM has shown superiority over standard AI computations with the highest scores in terms of understanding, specificity, and g-mean.

M e r c a l d o and S a n t o n e [19] Contemporary and exploratory networks provided a few ways to overcome the drawbacks of the progress signature-based positioning methodologies adopted by free and businesses antagonistic to malware. Most of these tactics are AI-managed, and to generate excellent predictive models, a perfect class of the ideal state is necessary. In this research, we use formal identicalness testing to identify the having a place family and present a method to derive reusable application malicious behavior.

Here the authors introduce new algorithms to decrease the number of flexible application tests and establish a metric to measure application harmfulness. Valid experiments conducted on 35 Android malware families between 2010 and 2018 confirm the practicality of the suggested method for flexible malware identification and family tree building. V u and J u n g [20] their analysis of AdMat is peculiar since it creates a contiguousness lattice for each application. These grids function as "input pictures" for the Convolutional Brain Organisation model, enabling it to distinguish between categories of malware and safe and harmful apps. We found that AdMat could adapt to different preparation sizes and achieve an average detection rate of 98.26% across multiple malware datasets throughout the evaluation period. With a predetermined quantity of preparatory data, it also successfully identified over 97.00% of different malware strains in implementation tasks.

G o n g et al. [21] show that developing such frameworks successfully involves several steps, including feature selection and encoding, designing and being transparent, application research efficiency and sufficiency, architect and client determination, and ML development of models. The "wooden barrel influence" of the whole structure could be triggered by discontent with any of the aforementioned points of view. To construct a workable ML-controlled malware identification system, the following paper outlines our rational strategy choices and first-hand organizational experiences. It has been operational at T-Market, checking approximately 12K applications every day with a single software computer. It has achieved a 98.9% overall accuracy and a 98.1% review rate, with an average of 0.9 min for each application analysis phase.

Here authors Y u a n et al. [22], essentially use a single computation for model construction. As a result, it might very easily be made entirely or gradually using a cell phone. When all factors are included, our indicator outperforms models that utilize shallow learning, such as Support Vector Machines (SVM) and AdaBoost, and gets close to models based on extensive learning, such as Multi-Facet Perceptrons (MLP) and Convolutional Neural Organization (CNN). Moreover, our value appears more robust against adversarial methods than the existing detectors, and its strength can be enhanced even more by upgrading on-gadget methods. Finally, extensive testing confirms the advantages it provides, and operational evaluation on smartphones demonstrates its rationality.

L i u et al. [23] presentation of extensions to previous investigations looks at a wider range of point components. In light of AI, this study provides a comprehensive summary of methods for discovering Android malware. For the time being, we briefly cover the basics of application development for Android, such as the technology behind the framework that makes up Android, safety standards, and the classification of Android malware. Next, with AI as our focus, we dissect and summarize the state of investigation from several angles, including test acquisition, knowledge pre-processing, highlighting identification, AI models, computations, and breakthrough sustainability evaluation. Finally, they assess the potential future research directions for Android malware detection given AI. This investigation will help academics get a complete picture of Android malware detection in the context of artificial intelligence. It might then catalyze subsequent researchers to start innovative initiatives and contribute to generally directing further investigation in this domain.

Here author D. L i and Q. L i [24] gathering advancing usually involves defensive measures, enemies might also utilize this process to improve their attack appropriateness. This motivates us to investigate the level of heartiness that the ensemble attack or ensemble protection can achieve, particularly during combat. Therefore, we suggest an alternative attack strategy called multifaceted incidents, which involves providing attackers suitable for multiple generation techniques and controlling sets to agitate malware architecture despite eliminating its harmful usage. This typically results in the introduction of extra disruptive training, which is also designed to enhance the collection of deep brain structures. We evaluate defenses with Android malware identifying characteristics versus 26 different attacks on two realistic datasets. Experimental results demonstrate that the new maladaptive training effectively increases the robustness of deep neural networks against numerous attacks; collection techniques improve robustness when the base classifiers are robust enough, but group attacks can evade the enhanced malware detectors and even significantly reduce the VirusTotal leadership.

3. Proposed architecture

This proposed research work has offered a novel approach to anticipate how malware in Android applications may function with an African Vulture Optimization-based Decision Tree (AVO-DT). Here the AVO-DT is applied to this algorithm and uses an optimization strategy that can get beyond the difficulties given by various optimization challenges as well as an ambiguous search space in multidimensionality.

The AVO-DT was inspired by This work and proposes a different algorithm inspired by African vulture's lifestyles with a comprehensive model to develop a new metaheuristic optimization algorithm., particularly simulates African vultures' foraging and navigation behaviors. The design of the AVO-DT is an attempt to create an algorithm that is quick, reliable, and user-friendly using AVO filled with adequate ability to capitalize on and investigate the realm of possibilities within alert modeling of the representative cum interactive features of the cape of vulture, also known as African vulture, in their search for clarifications [25]. AVO-DT replicates the cooperative behavior, communication skills, and group decision-making processes of African Vulture, which places great value on using the foraging and navigation hunting behaviors collective intelligence.

The development stages of the AVO-DT include the design of a metaheuristic optimization algorithm in twelve steps, as previously noted. These procedures were meticulously followed during the creation of the African Vulture Optimization.

Step 1. Initialize Vultures

def initialize_vultures(population_size):

return [np.random.rand() for _ in range(population_size)]

Step 2. Define Features (Assuming 10 features for demonstration)

num_features = 10

Step 3. Prepare Data (Assuming X as feature vectors and y as labels)

Replace X and Y with your actual dataset

X = np.random.rand(100, num_features)

y = np.random.randint(2, size=100) # Binary labels for demonstration

Step 4. Optimization Loop

num_generations = 10

population_size = 5

for generation in range(num_generations):

Step 5. Fitness Evaluation

fitness_scores = initialize_vultures(population_size)

Placeholder

Step 6. Selection and Reproduction (Not implemented for simplicity)

Step 7. Decision Tree Construction decision_trees = [] for vulture in range (population_size):

Assuming X_train, X_val, y_train, y_val are split data for training and validation

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

Step 8. Training

decision_tree = DecisionTreeClassifier()

decision_tree.fit (X_train, y_train)

Step 9. Evaluate on Validation Data

y_pred = decision_tree.predict(X_val)

accuracy = accuracy_score(y_val, y_pred)

fitness_scores[vulture] = accuracy

decision_trees.append(decision_tree)

Step 10. Survival of the Fittest (Not implemented for simplicity)
Step 11. Result
best_decision_tree=decision_trees[np.argmax(fitness_scores)]
Step 12. Application
Replace X_test with your actual test data
X_test = np.random.rand(10, num_features)
y_pred_test = best_decision_tree.predict(X_test).

4. Practical work and results discussion

The use of optimization issues to support operational procedures is discussed in this section. In terms of factor and random selection, they can evaluate the effectiveness of unmarked search algorithms as a result. After being used in a Python context, the function is tested. The software is developed in Python and is powered by an Intel Core i7 (1.8 GHz) CPU with 16 GB of RAM.

Testing process of the proposed AVO-DT Algorithm. By employing AVO decision variables to reframe the existing design issues, it would be possible to improve an algorithm. The optimization algorithm is constructed by re-creating the current design competition using AVO decision variables. We look at the solutions to well-known design problems. While creating a global optimization space for a problem, normal variables are swapped out with changeable variables. The effectiveness of the AVO-DT method in upgrading the AVO Algorithm was assessed.

4.1. Data sources description

For the training and testing phases of this model, the first dataset inputs are gathered. To carry out the training and testing process, both the dataset for safe Android applications and the dataset for well-known malware are gathered. So, the 5000 programs that are directly downloaded from the Google Play store are included in the benign Android application dataset that is used for processing. These programs serve a variety of purposes as well, including social media, talking, cooking, music, games, and education. The malware dataset also contains 5000 known samples. Thus, the malware dataset and the dataset for Android applications are used for processing during training and testing. The Android file format is identified as APK or Android Application Package. As a result, the suggested African Vulture Optimization-based Decision Tree (AVO-DT) model is processed on the dataset after the gathered dataset is taught to the system.

4.2. Comparative performance metrics evaluation with existing algorithm

Our objective is to suggest upgraded algorithm performed better than the alternatives. To illustrate the relevance of the results, a statistical test was conducted. To evaluate optimization problems and processes various top models are chosen using these standards. To completely evaluate the detection effect of Android malware, a variety of KPIs are typically combined. The proposed AVO-DT approach and the current AVO method are statistically compared in this paper. The performance of the developed model is compared to that of existing methods, such as the categorization

of harmful apps using images and fine-tuned neural models are TaintDroid [26], DroidScope [27], AdDroid [28], (IMCFN) [29], and SEdroid [30].

The following list includes the outcomes that were not considered in performance reviews. The accuracy, precision, recovery rate, and error precision of algorithms for optimization issues are evaluated using various ways.

4.3. Accuracy formulation

The performance of the AVO-DT model is computed depending on malware detection accuracy. To estimate reliable malware prediction, the AVO-DT model is used to compute detection accuracy as mentioned in the next equation:

$$\text{Accuracy} = \frac{\text{TPR}+\text{TNR}}{\text{TPR}+\text{TNR}+\text{FPR}+\text{FNR}}. \quad (1)$$

Table 1. Parameters evaluation of detection accuracy

Number of APKs in the dataset	Accuracy detection (%)					
	IMCFN	SE droid	Ad droid	Taint droid	Droid scope	AVO-DT (proposed)
1000	98.70	97.46	98.82	95.62	99.11	96.74
1500	98.48	98.72	98.56	92.56	98.67	97.31
2000	97.96	97.83	95.87	90.56	97.41	98.92
2500	98.54	98.72	96.21	93.67	95.39	98.5
3000	95.57	97.10	97.29	92.84	94.36	97.42
3500	98.64	98.72	98.38	99.33	98.34	98.68
4000	98.64	97.27	97.74	98.21	97.74	97.92
4500	98.64	98.30	96.48	95.36	96.48	97.05
5000	98.64	97.83	98.31	99.33	98.30	98.56

The presented AVO-DT model has higher detection accuracy than other models when compared to existing methodologies such as TaintDroid [26], DroidScope [27], AdDroid [28], (IMCFN) [29], and SEdroid [30], which is shown in Table 1.

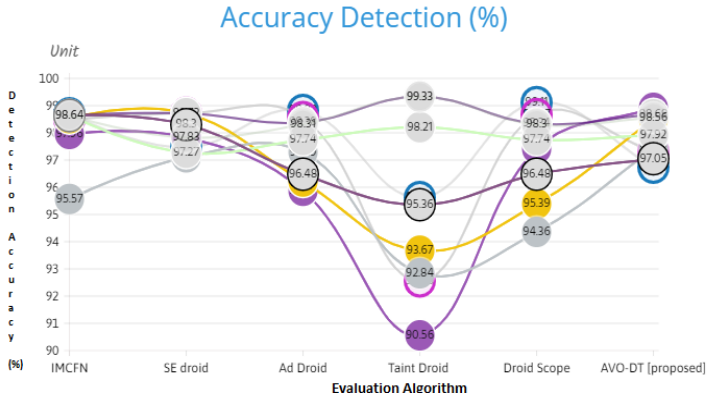


Fig. 1. Evaluation of detection accuracy

AVO-DT model demonstrated great malware detection accuracy when compared to other existing malware detection techniques. The constructed model achieved 99.85% detection accuracy when analyzing 5000 APK files in the dataset. Fig. 1 also shows the validation results.

4.4. Recall formulation

The proposed Enhanced Decision Tree Algorithm's detection ratio the AVO-DT model is considered based on the performance of the malware detection system in terms of recall. It is an estimate of the total number of malware programs accurately detected in the dataset that was used. Furthermore, it is utilized to calculate whether or not the suggested approach correctly detected the infection using the next equation:

$$\text{Recall} = \frac{\text{FPR}}{\text{TPR+TNR}}. \quad (2)$$

Table 2. Parameters evaluation of recall

Number of APKs in the dataset	Recall (%)					
	IMCFN	SE droid	Ad droid	Taint droid	Droid scope	AVO-DT (proposed)
1000	98.92	95.87	98.86	98.26	99.13	98.21
1500	98.37	96.21	98.67	97.75	98.44	98.82
2000	97.74	97.74	97.59	96.50	97.49	97.41
2500	96.48	98.38	98.34	92.44	98.89	96.02
3000	97.53	95.82	95.87	90.52	97.83	98.51
3500	96.48	96.21	96.21	93.67	95.41	98.46
4000	97.74	97.29	98.59	92.56	98.73	98.97
4500	96.48	98.38	95.87	90.56	97.56	98.77
5000	97.74	95.87	96.21	93.67	95.61	99.82

The demonstrated AVO-DT compared to other current models, such as IMCFN, SEdroid, AdDroid, TaintDroid, and DroidScope, this proposed model has a better recall rate in the table-2. It has also been validated using methods from TaintDroid [26], DroidScope [27], AdDroid [28], (IMCFN) [29], and SEdroid [30].

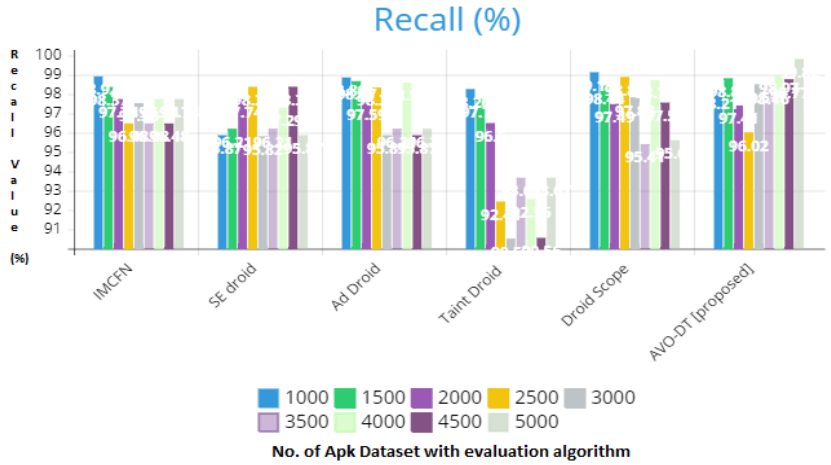


Fig. 2. Evaluation of recall value

The AVO-DT Model [31] fared better in terms of recall value than other malware detection algorithms currently in use. The developed model achieved a high recall value of 99.83% while taking into account 100 APK files in the dataset. Additionally, Fig. 2 displays the validation outcomes.

4.5. Precision formulation

It is the measure of attack detection confidence and is defined as the ratio of projections of positive results versus actual positive samples. It is discussed how to determine how many Android applications had positive detections in the entire dataset as

$$\text{Precision}=\frac{\text{TPR}}{\text{TPR+FPR}}. \quad (3)$$

Table 3. Parameters evaluation of precision

Number of APKs in the dataset	Precision (%)					
	IMCFN	SE droid	Ad droid	Taint droid	Droid scope	AVO-DT (proposed)
1000	98.6	97.59	96.5	98.3	99.33	98.06
1500	97.45	98.34	92.44	97.74	98.59	99.91
2000	98.46	95.87	90.52	96.48	95.69	99.40
2500	97.39	96.21	93.67	95.34	97.36	98.99
3000	97.08	98.59	92.56	94.76	94.35	98.46
3500	97.21	97.74	97.59	96.5	97.63	97.33
4000	96.48	98.43	98.48	92.44	98.42	96.43
4500	97.38	97.74	97.12	96.5	97.73	98.29
5000	96.29	98.46	98.76	92.28	98.49	98.69

The proposed AVO-DT model's accuracy is evaluated in comparison to current malware prediction techniques including TaintDroid [26], DroidScope [27], AdDroid [28], (IMCFN) [29], and SEdroid [30]., in Table 3.

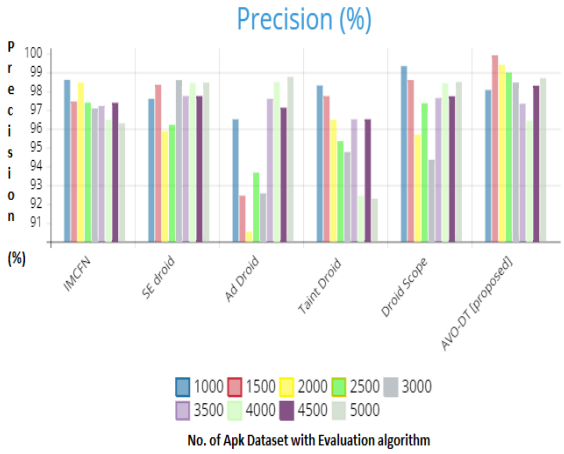


Fig. 3. Evaluation of precision

In comparison to other malware detection techniques currently in use, the AVO-DT models [31] with high precision values have been attained by the model. Assuming 100 APK files, the produced model has a high precision value of 99.76%. Additionally, Fig. 3 displays the validation results.

To assess the suggested AVO-DT technique based on the provided assessment requirements, the effectiveness analysis of several outcome parameters is computed.

The suggested approach based on the AVO Algorithm is well evaluated for a novel meta-heuristic optimization technique called the AVO-DT. Based on the aforementioned factors and the capability of the input data set to identify assaults, the proposed model is evaluated. The degree to which each incident was accurately detected was measured by accuracy; the percentage of attacks that were detected by the classifiers was measured by detection rate; and the number of attacks that the model returned was measured by recall. How many returned attacks were effective is referred to as precision.

5. Conclusion

By effectively evaluating the deep learning algorithms under consideration and using a new metaheuristic algorithm that was developed in this study, the study's goal of reducing the dimensionality of the problem space for problem-solving in the visual world was achieved. To advise the optimal technique choice from the evidence using 5000 applied data, metaheuristics have several algorithms like AVO and other motivational algorithms. Here, it is presumed that there are 2000 mobile applications in total, 2000 of which are malware applications. The effectiveness of the suggested algorithm is assessed and contrasted with the AVO algorithms meta-heuristic. The reason is to develop a novel AVO-DT to identify malware activities in Android applications.

Here, the proposed algorithm explores about malware features, and Android apps are educated for predetermined specific machines and tasks. Furthermore, the application incorporates the AVO-DT reporting technique to find malware in each program through static and dynamic investigation. To detect malware APKs in Android APKs, the created approach has demonstrated accuracy, precision, and recall capabilities. As a result, the suggested AVO-DT model attains a detection accuracy of 99.85%.

References

1. T a l u k d e r, S. K., M. I. I. S a k i b, M. M. R a h m a n. Model for e-Government in Bangladesh: A Unique ID-Based Approach. – In: Proc. of International Conference on Informatics, Electronics Vision (ICIEV'14), May 2014, pp. 1-6.
2. T a l u k d e r, S., B. C a r b u n a r. When a Friend Becomes Abuser: Evidence of Friend Abuse in Facebook. – In: Proc. of 9th ACM Conference on Web Science, Ser. WebSci '17. New York, NY, USA, ACM, June 2017 (Online). **http://doi.acm.org/10.1145/3091478.3098869**.
3. T a l u k d e r, S. K., M. I. I. S a k i b, M. M. R a h m a n. Digital Land Management System: A New Initiative for Bangladesh. – In: Proc. of 2014 International Conference on Electrical Engineering and Information Communication Technology, April 2014, pp. 1-6.
4. T a l u k d e r, S., I. I. S a k i b, F. H o s s e n, Z. R. T a l u k d e r, S. H o s s a i n. Attacks and Defenses in Mobile IP: Modeling with Stochastic Game Petri Net. – In: Proc. of 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC'17). IEEE, 2017, pp. 18-23.
5. L i, C., et al. Backdoor Attack on Machine Learning Based Android Malware Detectors. – In: IEEE Transactions on Dependable and Secure Computing, July 2021. DOI: 10.1109/TDSC.2021.3094824.

6. B l a z y t k o, T., M. C o n t a g, C. A s c h e r m a n n, T. H o l z. Syntia: Synthe-Sizing the Semantics of Obfuscated Code. – In: Proc. of 26th USENIX Security Symposium (USENIX Security 17), 2017, pp. 643-659.
7. Z h u, H., Y. L i, R. L i, J. L i, Z. Y o u, H. S o n g. SEDMDroid: An Enhanced Stacking Ensemble Framework for Android Malware Detection. – In: IEEE Transactions on Network Science and Engineering, Vol. **8**, 1 April-June 2021, No 2, pp. 984-994. DOI: 10.1109/TNSE.2020.2996379.
8. K a k a v a n d, M., D. M o h a m m a d, D. A l i. Application of Machine Learning Algorithms for Android Malware Detection. – In: Proc. of International Conference on Computational Intelligence and Intelligent Systems (CIIS'18), Phuket, Thailand, 17-19 November 2018, pp. 32-36.
9. L o p e s, J., C. S e r r a o, L. N u n e s, A. A l m e i d a, J. O l i v e i r a. Overview of Machine Learning Methods for Android Malware Identification. – In: Proc. of 7th IEEE International Symposium on Digital Forensics and Security (ISDFS'19), Barcelos, Portugal, 10-12 June 2019, pp. 1-6.
10. A l z u b a i d i, A. Recent Advances in Android Mobile Malware Detection: A Systematic Literature Review. – In: IEEE Access, Vol. **9**, 2021, pp. 146318-146349. DOI: 10.1109/ACCESS.2021.3123187.
11. K a t o, H., T. S a s a k i, I. S a s a s e. Android Malware Detection Based on Composition Ratio of Permission Pairs. – In: IEEE Access, Vol. **9**, 2021, pp. 130006-130019. DOI: 10.1109/ACCESS.2021.3113711.
12. W a n g, Z., J. C a i, S. C h e n g, M. L i. DroidDeepLearner: Identifying Android Malware Using Deep Learning. – In: Proc. of 37th IEEE Sarnoff Symposium, Newark, NJ, USA, 19-21 September, pp. 160-165.
13. L i, C., et al. Backdoor Attack on Machine Learning Based Android Malware Detectors. – In: IEEE Transactions on Dependable and Secure Computing, July 2021. DOI: 10.1109/TDSC.2021.3094824.
14. G o n g, L., Z. L i, H. W a n g, H. L i n, X. M a, Y. L i u. Overlay-Based Android Malware Detection at Market Scales: Systematically Adapting to the New Technological Landscape. – In: IEEE Transactions on Mobile Computing. DOI: 10.1109/TMC.2021.3079433.
15. S u r e n d r a n, R., T. T h o m a s, S. E m m a n u e l. GSDroid: Graph Signal Based Compact Feature Representation for Android Malware Detection. – Expert Systems with Applications, 2020, 113581.
16. M a h i n d r u, A., A. L. S a n g a l. MLDroid – Framework for Android Malware Detection Using Machine Learning Techniques. – Neural Computing and Applications, 2020, pp. 1-58.
17. M a r t i n e l l i, F., et al. Model Checking and Machine Learning Techniques for Humming Bad Mobile Malware Detection and Mitigation. – Simulation Modelling Practice and Theory, Vol. **105**, 2020, 102169.
18. A l m o m a n i, I., et al. Android Ransomware Detection Based on a Hybrid Evolutionary Approach in the Context of Highly Imbalanced Data. – In: IEEE Access, Vol. **9**, 2021, pp. 57674-57691. DOI: 10.1109/ACCESS.2021.3071450.
19. M e r c a l d o, F., A. S a n t o n e. Formal Equivalence Checking for Mobile Malware Detection and Family Classification. – In: IEEE Transactions on Software Engineering. DOI: 10.1109/TSE.2021.3067061.
20. V u, L. N., S. J u n g. AdMat: A CNN-on-Matrix Approach to Android Malware Detection and Classification. – In: IEEE Access, Vol. **9**, 2021, pp. 39680-39694. DOI: 10.1109/ACCESS.2021.3063748.
21. G o n g, L., et al. Systematically Landing Machine Learning onto Market-Scale Mobile Malware Detection. – In: IEEE Transactions on Parallel and Distributed Systems, Vol. **32**, 1 July 2021, No 7, pp. 1615-1628. DOI: 10.1109/TPDS.2020.3046092.
22. Y u a n, W., Y. J i a n g, H. L i, M. C a i. A Lightweight On-Device Detection Method for Android Malware. – In: IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. **51**, September 2021, No 9, pp. 5600-5611. DOI: 10.1109/TSMC.2019.2958382.
23. L i u, K., S. X u, G. X u, M. Z h a n g, D. S u n, H. L i u. A Review of Android Malware Detection Approaches Based on Machine Learning. – In: IEEE Access, Vol. **8**, 2020, pp. 124579-124607. DOI: 10.1109/ACCESS.2020.3006143.

24. L i, D., Q. L i. Adversarial Deep Ensemble: Evasion Attacks and Defenses for Malware Detection. – In: IEEE Transactions on Information Forensics and Security, Vol. **15**, 2020, pp. 3886-3900. DOI: 10.1109/TIFS.2020.3003571.
25. A b d o l l a h z a d e h, B., F. G h a r e h c h o p o g h, S. M i r j a l i l i, M. N. N o r a z i a h. African Vultures Optimization Algorithm: A New Nature-Inspired Metaheuristic Algorithm for Global Optimization Problems. – Computers & Industrial Engineering, 2021. DOI: 10.1016/j.cie.2021.107408.
26. E n c k, W., P. G i l b e r t, B. G o n C h u n, L. P. C o x, J. J u n g, P. M c D a n i e l, A. S h e t h. Taintdroid: An Information-Flow Tracking System for Real-Time Privacy Monitoring on Smartphones. – In: Proc. of USENIX Symposium on Operating Systems Design and Implementation (OSDI'10), 2010, pp. 393-407.
27. Y a n, L.-K., H. Y i n. Droidscope: Seamlessly Reconstructing OS and Dalvik Semantic Views for Dynamic Android Malware Analysis. – In: Proc. of USENIX Security Symposium, 2012.
28. M e h t a b, A., W. B. S h a h i d, T. Y a q o o b et al. AdDroid: Rule-Based Machine Learning Framework for Android Malware Analysis. – Mobile Netw. Appl., Vol. **25**, 2020, pp. 180-192. DOI: 10.1007/s11036-019-01248-0.
29. V a s a n, D., M. A l a z a b, S. W a s s a n et al. IMCFN: Image-Based Malware Classification Using Fine-Tuned Convolutional Neural Network Architecture. – Comput. Netw., Vol. **171**, 2020, 107138. DOI: 10.1016/j.comnet.2020.107138.
30. W a n g, J., Q. J i n g, J. G a o et al. SEdroid: A Robust Android Malware Detector Using Selective Ensemble Learning. – IEEE Wirel Commun Netw Conf., 2020, pp. 1-5. DOI: 10.1109/WCNC45663.2020.9120537.
31. K u m a r, P. K., V. S h a r m a. A Novel Efficient Optimized Machine Learning Approach to Detect Malware Activities in Android Applications. – Multimedica Tools and Application, Vol. **82**, April 2023.
32. G u l i a s h k i, V., L. K i r i l o v, A. N u z i. Optimization Models and Strategy Approaches Dealing with Economic Crises, Natural Disasters, and Pandemics – An Overview. – Cybernetics and Information Technologies, Vol. **23**, 2023, No 4, pp. 3-25.

Received: 12.01.2024; Second Version: 18.04.2024; Accepted: 31.05.2024