applied sciences

MDPI

Article

Machine Learning-Based Adaptive Genetic Algorithm for Android Malware Detection in Auto-Driving Vehicles

Layth Hammood [1,2,*], İbrahim Alper Doğru [1] and Kazım Kılıç [1]

[1] MobSecLab, Department of Computer Engineering, Faculty of Technology, Gazi University, Teknikokullar Ankara 06560, Turkey; iadogru@gazi.edu.tr (İ.A.D.); kazim.kilic@gazi.edu.tr (K.K.)
[2] College of Computer Science and Information Technology, Kirkuk University, Kirkuk 36001, Iraq
* Correspondence: lsattam.hammood@gazi.edu.tr; Tel.: +90-534-715-7997

Abstract: The growing trend toward vehicles being connected to various unidentified devices, such as other vehicles or infrastructure, increases the possibility of external attacks on"vehicle cybersecurity (VC). Detection of intrusion is a very important part of network security for vehicles such as connected vehicles, that have open connectivity, and self-driving vehicles. Consequently, security has become an important requirement in trying to protect these vehicles as attackers have become more sophisticated in using malware that can penetrate and harm vehicle control units as technology advances. Thus, ensuring the vehicles and the network are safe is very important for the growth of the automotive industry and for people to have more faith in it. In this study, a machine learning-based detection approach using hybrid analysis-based particle swarm optimization (PSO) and an adaptive genetic algorithm (AGA) is presented for Android malware detection in auto-driving vehicles. The "CCCS-CIC-AndMal-2020" dataset containing 13 different malware categories and 9504 hybrid features was used for the experiments. In the proposed approach, firstly, feature selection is performed by applying PSO to the features in the dataset. In the next step, the performance of XGBoost and random forest (RF) machine learning classifiers is optimized using the AGA. In the experiments performed, a 99.82% accuracy and F-score were obtained with the XGBoost classifier, which was developed using PSO-based feature selection and AGA-based hyperparameter optimization. With the random forest classifier, a 98.72% accuracy and F-score were achieved. Our results show that the application of PSO and an AGA greatly increases the performance in the classification of the information obtained from the hybrid analysis.

Keywords: Android malware detection; machine learning; auto-driving; particle swarm optimization; adaptive genetic algorithm

Citation: Hammood, L.; Doğru, İ.A.; Kılıç, K. Machine Learning-Based Adaptive Genetic Algorithm for Android Malware Detection in Auto-Driving Vehicles. *Appl. Sci.* **2023**, *13*, 5403. https://doi.org/10.3390/app13095403

Academic Editor: Xianpeng Wang

Received: 5 February 2023
Revised: 9 April 2023
Accepted: 21 April 2023
Published: 26 April 2023

1. Introduction

Globally, as computers and the internet have spread more widely, the security threat landscape has also undergone a significant transformation from sporadic and unorganized attacks with a primary focus on fame to more coordinated and multi-vector attacks with a global scope and a financial profit motive [1,2]. On the other hand, the development of transportation networks has increased just as it has increased in auto-driving vehicles, as has the expansion of vehicle connectivity and the spread of vehicle communication as a result of the tremendous growth in technological development [1]. Computers are compromised; networks are breached; private information is stolen; tons of spam emails are sent; servers are brought down; and vital infrastructures are crippled; all of which cause serious damage and substantial financial losses [1,2]. Due to the limited level of protection on computers used by common users and the high value of business targets, skilled and motivated cybercriminals have launched various security attacks.

A recent Computer Security Institute (CSI) survey found that each security attack resulted in an average loss of about $345,000 [3].

In recent years, general trends in threats have emerged, and millions of "malicious programs" have targeted individuals, while the frequency and quality of these attacks have evolved. Malware infection rates on Android devices have increased by 400% [4]. In addition, malware of all kinds (malicious software) is the predominant transportation method for a huge percentage of structured cybercrimes. It is intended to infiltrate computer networks and also manipulate them for malicious ends. Viruses, worms, spyware, Trojan horses, rootkits, and bots are examples of common malware types [2]. They spread a destructive payload, infect vulnerable computer systems, take over control over them, and use them as a platform for additional criminal activity and make money illegally [5]. Malware can be very challenging to detect and identify because it has attributes such as many configurations, complex attitudes, and sophisticated secretive techniques, all of which analyze dangerous codes which can be challenging and time-consuming [3]. As a result, anti-virus engineering is something that both developers and users need to be aware of. Future cyberattacks on automobiles will rise due to the increased use of communications [1]. Therefore, to defend against cyberthreats, countermeasures must be implemented for every attack technique [2,3].

Cybercriminals and malware programmers frequently target self-driving automobiles/vehicles. It is well known that connected vehicles operate in a confined environment and can only receive remote control commands through approved closed communication channels, such as a server made by the manufacturer or special software [6]. A confined environment prevents the issuance of malicious or unauthorized commands [1,2]. On the other hand, modern autonomous vehicles can communicate with other vehicles, infrastructure, and intelligent devices outside the vehicle via control signals and internal data [2]. The vehicle's security is crucial at this time, so it is necessary to examine all attack vectors against automobiles through safe and private cybersecurity. The safety of self-driving vehicle occupants is intimately linked to the vehicle's security [5].

Researchers have tried to stop people from undertaking things they should not undertake by installing gates that allow only legitimate and approved procedures by using systems to look for bad behavior. Still, this procedure was not enough to stop them before they happened. It is very important to use data analysis technology to examine situations that are not normal and see if any behaviors have not been observed before [3]. Android malware poses various security problems for self-driving vehicles, and this study aims to find a way to identify such malware as a threat. Due to resource constraints and requirements for real-time responses in an integrated environment, such as autonomous and connected vehicles, detection accuracy and reaction time are essential variables. They must be considered [6].

There are some important aspects to consider. Firstly, the process of communication from a vehicle to a device (V2D) is performed by transmitting information from inside the vehicle to the outside world by connecting to a server or a smartphone [5]. Therefore, the smartphone can be used for automotive software upgrades where the smartphone is not limited to playing music or other traditional things. On the other hand, this type of communication between the vehicle and the smartphone can be dangerous to the vehicle's network if someone hacks into the connection or the phone is infected with malware. Secondly, at the level of applications using malicious code, Android-based hacking attacks on Android devices are one of the most common attack methods [6]. This occurs when an application carrying malicious code that may have been downloaded via email or a web page is utilized without the knowledge of the user as the device is injected with malicious software as part of a hacking attack [2,3].

A variety of interfaces are used to connect linked or self-driving autos to external or public networks [7]. Connectivity Control Units (CCUs) or Telematics Control Units (TCUs) are devices that have an external communication and modem that allow them to receive GPS signals and connect to wireless networks. Numerous apps for in-vehicle infotainment systems are made possible by embedded operating systems, such as Android OS or QNX OS. During the development of wireless or wired networks, and if security is not taken

into account, it is possible to take over the OS environment by using embedded malware or malicious instructions to perform evil jobs [7].

In general, the malware detection phase has mainly been divided into two phases in previous studies: the first one has been the analysis phase and the second one has been the detection phase. In the analysis phase, to extract the features of the malware, several methods are available such as static analysis (SA), dynamic analysis (DA), memory analysis (MA), and hybrid analysis (HA) [8]. Static analysis includes testing and analyzing specific malware without running it directly. Whereas dynamic malware analysis is carried out systematically and in a controlled environment to analyze the functioning of the malware. Hybrid analysis uses both dynamic and static analysis. The last method is the memory analysis method, which performs a complete and comprehensive analysis of the malware in the memory. When the malware analysis is completed, the discovery phase is started to detect the malware. Through the use of malware detection stages, it will be possible to prevent the spread of malware and prevent the damage that can occur because of that. Therefore, this paper is based on the static analysis of malware detection in self-driving vehicles [7].

There are many studies which have been performed using static and dynamic analysis for Android malware detection [9–12]. Static analysis is fast and the application can be avoided without damaging the system. However, the static analysis method is vulnerable to zero-day attacks. It is also vulnerable to runtime vulnerabilities. Dynamic analysis, on the other hand, is powerful in terms of detecting and capturing the real-time behavior of the application. In this study, a dataset containing hybrid analysis was used to take advantage of the detection power of static analysis and dynamic analysis, and the experiments were carried out with hybrid analysis. In the current studies, high accuracies have tried to be obtained by using different analysis types and different classification models have been presented. However, machine learning and deep learning methods presented in previous studies are dependent on hyperparameter selection. In addition, selecting the best features requires expert knowledge. In this study, the best features are selected by using particle swarm optimization on the dataset containing hybrid analysis information. At the same time, automatic adjustment of hyperparameter settings is provided using the adaptive genetic algorithm.

The specific and significant contributions of this current study can be summarized as follows:

- In the study, a dataset with a wide hybrid feature is used for the security of Android systems used in autonomous vehicles. In this way, the malware detection power of static analysis and dynamic analysis is utilized. Particle swarm optimization technique is used in order to discard irrelevant features and use related features.
- Adaptive genetic algorithm is used to maximize the performance of machine learning classifiers.
- The performance of the proposed AGA-based XGBoost and random forest classifiers is compared with deep neural network and various machine learning classifiers.
- The results obtained with the proposed method have reached more successful accuracy values than similar studies for Android malware detection.

The weaknesses of the existing studies are summarized below:

- Dynamic features are needed for the hybrid analysis method used in this study. Acquiring dynamic attributes is time-consuming.
- The selection and classification of features in the proposed PSO- and AGA-based detection approach requires expert knowledge. With deep learning methods, this is undertaken automatically.
- Experiments have been conducted in the local environment and no real-world testing has been conducted for the safety of autonomous vehicles.

The configuration of the current study is as follows: A review of the pertinent research is explained in Section 2 while a brief description is given of the intrusion detection module

based on machine learning in Section 3. The present study describes the entire Materials and Methods in Section 3 whereas Section 4 presents the methods for measuring the classifier performance. Section 5 presents the hyperparameters of the experiments. Observations regarding the experience and results/are provided in Section 6, it also consists of a discussion regarding this study's findings and in Section 7, the conclusions are explained.

2. Related Work

Machine learning techniques are included and applied in a variety of articles dealing with malware identification and categorization. Existing works use dynamic or static analysis as a way to extract features that use malware detection. For example, Nikola Milosevic et al. introduced two static analysis methods for Android malware [5]. The first strategy relies first on permissions, while the second method relies on parsing the source code aided by the application of machine learning which uses a model that represents a bag of words to describe the data. The F-score for source code-based classification models was 95.1%, while the F-measure for permission-based classification models was 89% [6].

Thomas et al. redirected all malicious processes from the hacked device to a limited virtual machine via installation techniques, so it is a defensive solution for malware. In addition, they returned the results of the malicious behavior to the malware as if it had been performed on the real device [13].

Iqbal et al. suggested the classification and protection of vehicles from malware. They also offered a security architecture to keep vehicles safe. They used virtualization to make it less vulnerable to attacks. All vital functions of the vehicle were controlled by a real-time operating system, while at the same time, several additional operating systems were used to control non-critical functions of the vehicle (information and entertainment, information technology, etc.) [14].

Seunghyun Park et al. introduced a method that relies on machine learning to detect all the anomalies caused by malware in real-time broadband network traffic. After that, the researchers and scientists presented an effective and essential algorithm which involved identifying malicious actions within the network environment, and they also performed experiments that verify the accuracy of the suggested algorithm and make comparisons with new technologies [9].

Zhang et al. presented a TC-Droid to utilize the on-text categorization algorithm to automatically find malware on Android devices. The suggested system takes the script sequence of AndroPyTool application analysis reports and uses a CNN to analyze all relevant information in the original report, rather than manually engineering features [10].

Daniel et al. extracted features in the source code and apps' manifest files and used them to make eight feature sets [12]. A support vector machine classification algorithm related to malware detection was trained by utilizing this method, and a relatively high rate of malware family detection was obtained when employing this classifier [15].

For better virus detection on Android IoT devices, Rajesh Kumar et al. developed a novel framework combining machine learning and blockchain technology. Sequential clustering, classification, and blockchain are all used in the suggested technique. Machine learning removes malware data automatically and stores them in the blockchain. Thus, the network can broadcast all malware data stored in the blockchain history, enabling excellent malware detection [16].

Seungho Jeon et al. proposed a model in which opcode-level convolutional autoencoders compress long opcode sequences, and dynamic recurrent neural network classifiers execute prediction tasks using the codes generated by the opcode-level convolutional autoencoder [17].

3. Materials and Methods

Machine learning (ML) relies on a set of algorithms used to generate results that aid decision-making. These ML algorithms consist of three types [18] which are unsupervised learning (UL), supervised learning (SL), and reinforcement learning (RL). The

most prominent algorithms commonly used are random forest (RF), Naive Bayes (NB), k-nearest neighbor (KNN), support vector machine (SVM), logistic regression (LR), artificial neural network (ANN), gradient boosting (GB), extreme gradient boosting (XGBoost) and adaptive genetic algorithm (AGA).

3.1. Methodology

The steps of the current study's methodology are illustrated in Figure 1. Initially, a CCCS-CIC-AndMal-2020 dataset that had been created by the Canadian Institute for Cybersecurity (CIC), in 2020, was selected. After collecting the data, we processed them to make them more clear and rich in information for use in our proposed method and machine learning algorithms through which the data were cleaned by removing incorrect, damaged, and duplicate data within the user dataset. Feature selection methods aim to reduce the number of input variables to those believed to be most useful for the models to predict the target variable to improve the models' performance. Input variables that have the strongest relationship with the target variable were chosen. The choice of statistical measures depends on the type of data for each of the input and output variables. Therefore, for feature selection, we used a PSO algorithm with cross-entropy and KL divergence. This helps our proposed method to select the best features, as shown in Figure 2. After completing the data processing for the used dataset, the data were segmented into a training set of 80% and a test set of 20%. Training data will be passed to the AGA for tuning the hyperparameter of the random forest and Naïve Bayes. F1-score; precision; recall; accuracy; ROC curve; and confusion matrix were used to assess the efficiency of the classifiers in the set of data.

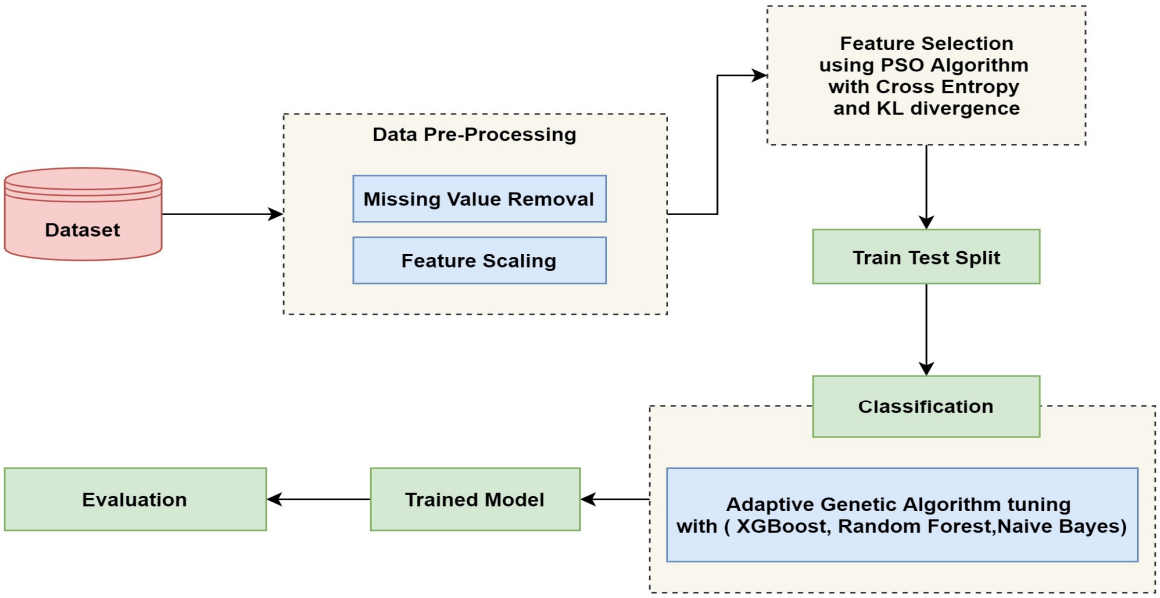


Figure 1. The proposed methodology.

3.2. Dataset

In the current research study, the CCCS-CIC-AndMal-2020 dataset developed by the Canadian Institute for Cybersecurity (CIC) [19,20] was employed. The CCCS-CIC-AndMal-2020 dataset is widely acknowledged in the academic literature and has served as the foundation for a variety of studies. Static analysis data from the CCCS-CIC-AndMal-2020 dataset are used, which include 7000 benign and 7000 malicious Android samples, as shown in Table 1. The malicious samples consist of 13 categories which are Trojan, Zeroday, Ransomware, SMS, Spy, Scareware, PUA, NoCategory, Dropper, Banker, FileInfector, Backdoor, and Adware, as shown in Table 2. The dataset utilized has 9504 features. We split the dataset into two partitions, 80% for training and 20% for testing.

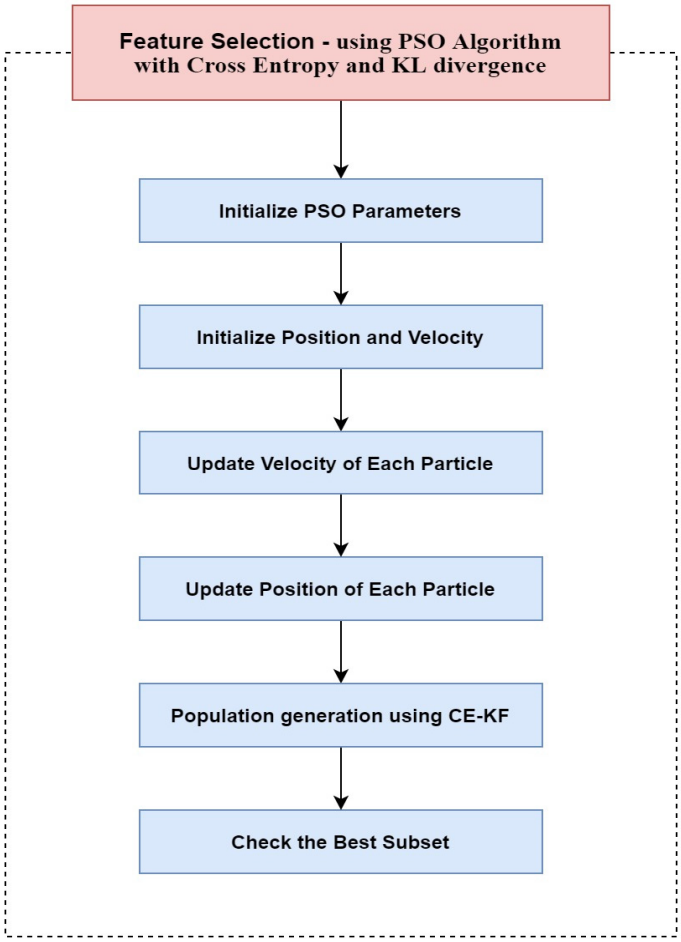


Figure 2. Feature selection.

Table 1. Showing datasets types.

Dataset	Number of Datasets	Ratio
Benign	7000	50%
Malware	7000	50%

Table 2. Showing malware types.

Malware Type	Number of Samples
Trojan	500
Zeroday	500
Ransomware	500
SMS	500
Spy	500
Scareware	500
PUA	500
category	500
Dropper	1000
Banker	500
FileInfector	500
Backdoor	750
Adware	250

3.3. Features Selection Using PSO Algorithm with Cross-Entropy (CE) and Kullback–Leibler (KL) Divergence

As previously indicated, the dataset contains 9504 features. In this phase, we used the choose features technique to select the most essential features that have a high correlation to the type of app that we want to forecast. Choosing a set of important features from the dataset helps greatly in reducing the time to test algorithms. Some features may be unnecessary or redundant, so they are excluded to help the accuracy of the prediction and the speed of analysis of the results. Therefore, for feature selection in our proposed method, we used a PSO algorithm with cross-entropy (CE) and Kullback–Leibler (KL) divergence.

The KL divergence involved measuring the CE between two sampling distributions and solves an optimization problem by minimizing them and obtaining the optimal probability distribution parameters. CE along with KL divergence has the few important advantages of strong global optimization ability, good adaptability, and strong robustness. Our proposed meta-heuristic PSO algorithm with CE with KL divergence method for feature selection is embedded in the PSO algorithm to provide a good balance between exploitation and exploration. Based on this algorithm, we gain the optimal subset from a given set of features in the dataset, as shown in Figure 2.

3.4. Classification Methods

Support vector machine (SVM) is a widely used machine learning method for classification problems; because of its flexibility and simplicity properties, that are used in addressing classification problems, the SVM method provides a characteristic and balanced predictive performance that is needed to perform well during studies with potentially small sample sizes. It also has the potential to help solve big data classification problems. Therefore, the SVM method has proven to be an efficient and flexible tool for classification and analysis [21].

SVM is a type of supervised machine learning algorithm classifier that is utilized for the examination of data that is also involved in the identification of various patterns which are used for the classification. Between a series of patterns of various classes, this classifier creates a hyperplane. The SVM linear classifier's mathematical form is displayed in (1).

$$f(x) = WT\,X + b \tag{1}$$

Random forest (RF) is a classification and regression technique which depends on the aggregation of several decision trees. RFs are used to analyze and classify quantities of data. Specifically, it is a set of trees used to build a training dataset and conduct an internal evaluation in order to arrive at a reasonable prediction of the response given future predictions. This is undertaken so that the model can be used to make predictions [22].

Logistic regression (LR) analysis can be defined as one of the statistical methods that are based on evaluating the relationship between all different prediction variables, whether they are definitive or continuous. When there is more than one variable, LR is used to obtain the probability ratio. LR works similarly to linear regression to a large extent, but the difference between them is that logistic regression works with a variable whose response is binomial. To reduce all the effects of complex factors, LR analyzes all multiple variables simultaneously, so the LR algorithm is a very powerful tool [23].

The decision tree's (DT) structure is the same as that of a regular tree, which consists of roots, branches, and leaves in the internal nodes where the attribute is tested. In the branches, the test result is in it, and in the leaf node, the test result is labelled. DT algorithms use attributes and segmentation to test which nodes are better at segmenting into individual classes. The training data are divided into subgroups according to the values of the segmentation attributes. The algorithm continues until all the similarities are in a group of the same class [24].

K-nearest neighbor (KNN) is one of the most straightforward ML methods for solving classification issues. The notion behind how the algorithm is supposed to function is that items that are close to each other also have attributes that are close to and comparable to

each other. The demand for a considerable amount of memory to store the entire sample is one of the primary drawbacks of using KNN classifiers. When the sample size is huge, the reaction time on a sequential computer likewise increases significantly. Despite this issue, it still performs satisfactorily. Classifiers benefit from their improved performance because it helps them solve classification difficulties [25].

Artificial neural networks (ANNs) are utilized in dataset analysis and classification, while they can also be involved in the identification of recognized patterns and make predictions. When available, ANNs can analyze massive data and produce very precise predictions. The most common and used ANNs are connected and also supervised networks [26].

The Bayes theorem is the basis for the Naive Bayes (NB) classification method, which is a simple probabilistic classification approach. It also operates on the presumption that each feature is distinct from the others. The model must first be instructed using the training dataset in order to produce reliable predictions using the predict function. Although it is only a sample approach in ML methodologies, it has the potential to be effective in some situations. The training is simple and quick; all that is required is to take into account many predictors for each class on its own [27].

Gradient boosting is a machine learning method that uses a commonly used boosting technique in classification tasks and regression problems. Boosting is a type of batch learning method that trains the model sequentially and each new model tries to fix the previous model. In this method, weak learners are combined into strong learners. In this process, each new model is trained to minimize the loss function such as mean square error or cross-entropy of the previous weak model by using a gradient descent technique. At each iteration, the GBM algorithm calculates the gradient of the loss function based on the current group's estimates and then trains a new weak model to minimize this gradient. The prediction of the newly trained model is added to the ensemble model and iteration continues until the stopping criterion is met [28].

XGBoost is a modified and optimized high-performance version of the gradient boosting algorithm. It was introduced in the article "XGBoost: A Scalable Tree Boosting System". The most important features of this algorithm are that it can achieve high predictive power, prevent over-learning, manage empty data, and perform them quickly. According to the creators of the algorithm, XGBoost works 10 times faster than other popular algorithms.

In the XGBoost algorithm, software and hardware optimization techniques have been applied to obtain superior results by using less resources than its predecessors. It has been shown to be the best of the decision tree-based algorithms and boosting methods [29]. In addition, XGBoost is optimized for working with big data.

3.5. Particle Swarm Optimization Methods (PSO)

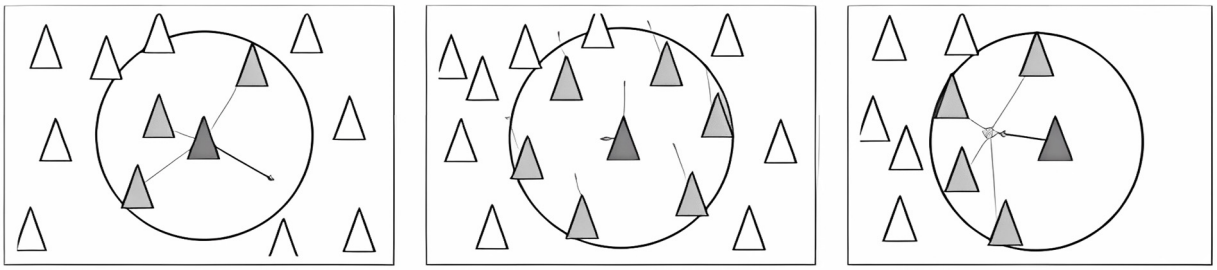
In 1995, Kennedy and Eberhart released the first version of the particle swarm optimization (PSO) technique. PSO techniques are population-based optimization algorithms that guide particle search for universally optimal solutions by simulating interactions between flocks of birds and schools of fish. PSO, an unsupervised algorithm, is used to find more efficient ways to extract features. PSO involves group members interacting with one another. Based on their individual research and the group's collective data, each member modifies the group's course. When using PSO to solve an optimization problem, a swarm of computer parts called particles scour the solution space in search of the best solution [30].

The PSO algorithm starts with a population of random particles (swarms) and then updates generations to look for the best answer. The present position and velocity vectors are connected to each particle. These vectors are the same dimension as the search space in terms of size. The two best values are updated for each particle in every iteration. The best fitness value is found in the swarm, and the other is which stores the fitness value of the greatest fitness solution [31]. However, Del-Valle and his co-authors defined PSO as having the three simple behaviors of separation, alignment, and cohesion (Figure 3a–c, respectively). Separation is the behavior of avoiding the crowded local flock mates, whereas

alignment is the behavior of moving in the general direction of the local flock mates. The PSO algorithm's formulas are as follows [32], and cohesion is the behavior of moving toward the median position.

$$\text{vid t} + 1 = \text{vt td} + \text{c 1. rand} - (0,1).(\text{pt td} - \text{xttd}) + \text{c 2. rand} - (0,1).(\text{ptgd} - \text{xttd}) \quad (2)$$

$$\text{xid t} + 1 = \text{xtid} + \text{vidt} + 1 \quad (3)$$



(**a**) (**b**) (**c**)

Figure 3. (**a**) A separation behavior in which particles try to avoid coming into contact with one another. (**b**) Alignment. behavior, where particles move themselves with the leader of their local flock while keeping their relative velocities constant. (**c**) Cohesion. behavior, where particles tend to move to where their neighbors in the flock are.

Note: where, respectively, vt id and xt id represent particle position and velocity, it is the particle index;t which is the iteration number, and d is the dimension of the search space. When flying toward the most ideal individual particle and the ideal particle overall, c1and c2 stand for the speed that controls the length of the flight. Pi is the highest position that a particle has yet attained. The best position determined by the particle's neighbors is I and pg. I. Rand (0, 1) represents the values at random between 0 and 1. The differences between the population's all-time best (ptgd) and exploration happen when the difference between the position of the previous particle (xtid) and the position of the best particle (ptid) is large. When these two value systems are small, there is exploitation.

In addition, at the first step, the population is implemented through the PSO algorithm and then the second step involves determining each particle's fitness values which are updating the global and individual bests, then updating the position and velocity of the particles. Up until the termination, the condition is met, and steps two through four are repeated. The output of the PSO algorithm over iterations is shown in Figure 4. To find the best solution in the first iteration, all particles dispersed (exploration) and also every particle is assessed. The ideal answers are discovered regarding swarm members' neighborhood configuration, and their individual and collective best particles are updated. Convergence would be reached by trying to move all the particles toward the one with the best solution [32].

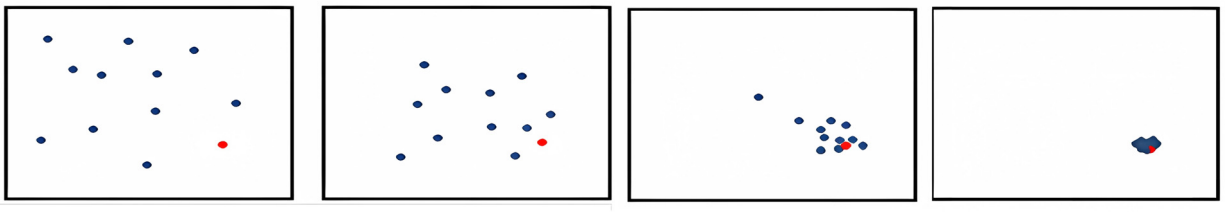


Figure 4. Over iterations, particle swarm optimization tends toward global optimums.

In recent times, the PSO algorithm has received a lot of attention due to its several benefits. In addition, it works well for global search and does not depend on the size of design variables. It is easy to set up and only needs a few parameters to be set; it can be easily parallelized for simultaneous processing. The PSO algorithm's strengths are as follows [33]:

1. It is very resilient and adaptable to many application contexts with minimal change.
2. It is easy to perform parallel computations because the algorithm is the swarm evolutionary algorithm. This gives it a strong ability to be spread out.
3. It can quickly move close to the optimal value.
4. It is simple to hybridize or combine with other algorithms to enhance its performance.

The PSO algorithm generates a swarm of particles at random locations to represent possible solutions to the optimization task. In order to achieve the optimal value, the particle positions are repeated until they reach the maximum of a fitness value. Then, the PSO checks the positions of each particle and stores the best solution for each particle. Furthermore, the PSO keeps a record of the best solutions at the end of each iteration. Particle positions and velocities are calculated in each subsequent iteration, using optimal swarm positions, the best position of the particle itself, and previous particle velocities. The computational process is carried out for 25 iterations. Based on the use of this algorithm, 4699 optimal features were obtained from the sum of features of the CCCS-CIC-AndMal-2020 dataset. Figure 5 shows how the particle swarm optimization shapes the optimization for feature selection.

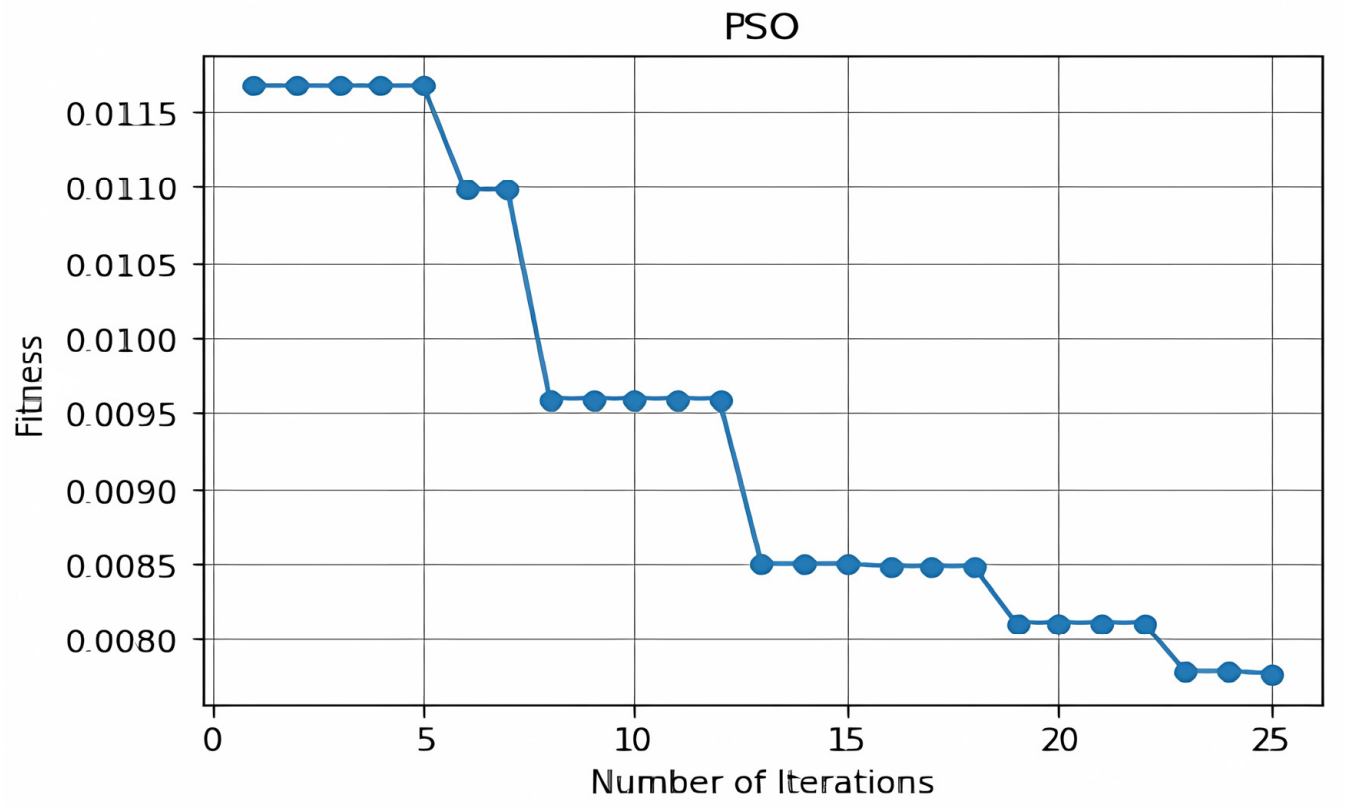


Figure 5. The convergence graph for particle swarm optimization.

3.6. Adaptive Genetic Algorithm(AGA)

The genetic algorithm (GA) is an efficient optimization tool that uses a computational model inspired by the principles of natural selection to solve problems [4]. In 1975, Holland envisioned a GA that was based on biological evolution which evolved through crossover and mutation processes to find the optimized solutions because genes in nature partially crossover, mutate to create new genes that differ from existing ones, and evolve into an

environment. As the generation time passes, the response that is acquired via the use of a GA converges in a particular fashion. Selecting the perfect combinations and enigmatic solutions are some of its benefits, along with securing and exploring potentially large search spaces. Consequently, it is appropriate for determining a solution to an NP-hard problem, such as feature selection [34].

A GA typically starts with randomly generated gene populations, according to D. Whitley. They evaluated the genes that have been assembled and distributed possibilities for every gene's upcoming generation, and offered the genes that met the criteria for the problem more chances for reproduction than those that did not. Then, just the genes that satisfied specific requirements were selected; they were then put together in a variety of different ways to produce new genes. However, crossovers have the potential to create new genes, and mutations have the potential to follow specific probabilities. Up until the algorithm's termination criteria are satisfied, this process is repeated. A simplified flowchart of these procedures is depicted in Figure 6 [35].

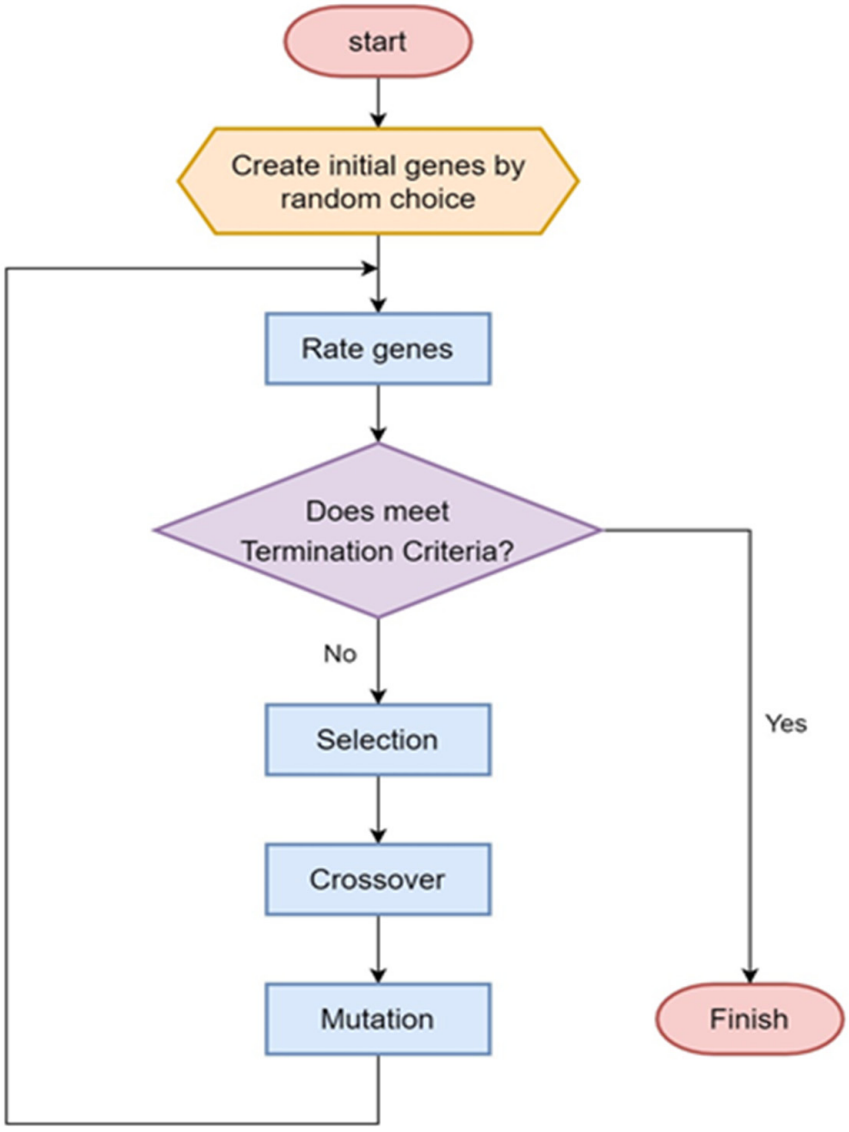


Figure 6. A schematic diagram to present the pattern of how the "genetic algorithm" progresses.

For all of the above and the importance of an AGA, we have created our proposed method that helps to improve the results of the algorithms that we have chosen. What helped our method produce excellent results is the selection of features by the PSO algorithm. The AGA has been used to adjust the hyperparameter through the AdaptiveGenet-

icSelectionCV method for three machine learning classifiers (RF, NB, and XGBoost). The appropriate parameters for this method were determined to obtain the best results, as we found when implementing and measuring the performance of the classifiers.

However, many versions of well-established GAs are presented as ways to enhance this technique. One such technique is the adaptive genetic algorithm (AGA). In the AGA, the chances of a crossing or mutation are weighted by fitness, allowing for more optimal results to be found. Coding, fitness computation, selection, reproduction, crossover, mutation, and decoding are all part of the AGA process. On the other hand, to increase the effectiveness of the GA, AGAs are searching for a more broadly applicable adaptive crossover and mutation probability. Both cross-over and mutation probability in the AGA is a dynamic adjustment between the population's average fitness and highest fitness level at each individual's fitness level. The following relationship is satisfied by them being displayed in (4) and (5):

$$P_X = \begin{cases} k1 = \dfrac{(fmax - f)}{fmax - favg}, & f \geq favg \\ k3, & f' < favg \end{cases} \tag{4}$$

$$P_m = \begin{cases} k2 = \dfrac{(fmax - f)}{fmax - favg}, & f \geq favg \\ k4, & f' < favg \end{cases} \tag{5}$$

Note: the largest fitness value for the group is *fmax*; the average fitness for the population is *favg*; the largest individual to cross the fitness is f [8]; and the mutation individual's fitness value is f. Simply set this up to take the value of $k1$, $k2$, $k3$, and $k4$ (in [0, 1] value), and P_c and P_m can be adjusted adaptively.

Each candidate's solution to the problem is set by the procedure, then evaluated using a fitness function. The best individuals would be kept around for the subsequent generation. Subsequently, the fitness is used to determine the chances for crossover and mutation in each successive generation. The AGA-based procedures will be applied, and then the result will be obtained. We will determine the crossover and mutation values based on the fitness value [29,30].

4. Evaluation

After preparing the data for the dataset, the experiment was performed using a Python application by Jupyter Notebook. In the present study, we assessed the proposed approach. Since the dataset is so big and complex, we attempted to reduce the training time by finding the best type and number of features using the PSO algorithm with CE and KL divergence to obtain a subset of features that have a high rate of detection. To evaluate the performance of our proposed method, a series of experiments were performed on the CCCS-CIC-AndMal-2020 dataset. Meanwhile, we have used a hybrid approach as a classifier to detect attacks with a high degree of accuracy [24]. The performance of the classifiers used in our method was measured by measuring the F1-score, precision, recall, and accuracy. The AUC, the area under the ROC curve, is a popular metric for assessing classifier performance in malware datasets with the following stated levels: 0.5 indicates a poor guess, 0.6 indicates a poor forecast, 0.7 indicates a mediocre forecast, 0.8 indicates a decent prognosis, 0.9 indicates an excellent forecast, and 1.0 indicates an excellent forecast. To summarize the performance of the algorithms that we used in our research paper, a confusion matrix was used in which the number of correct and incorrect predictions by each class is known, as shown below:

$$\text{F1} - \text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

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

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

5. Experiment

After conducting the experiments using the dataset, it was seen that the outputs of the project were promising. The algorithms gave satisfactory results and predictions on the test set. To avoid inadequate feature selection results, an AGA for hyperparameter tuning is presented. This algorithm performs parameter tuning for the classification algorithms RF and XGBoost. It is necessary to use AGA parameter tuning techniques with affordable computational costs. The main goal of hyperparameter tuning is to prevent early convergence and slow completion and to maximize the model's performance by minimizing a predefined loss function and resulting in better results with fewer errors. We came up with hybrid intelligent Android malware detection methods based on an evolving RF, referred to as (RF-AGA) and evolving XGBoost referred to as (XGB-AGA), which are both based on evolutionary algorithms. Experimental results show that the RF-AGA and XGB-AGA are significantly better than common ML classifiers and other malware detection methods.

6. Results and Discussion

In this section, the results of the experiments performed on the "CCCS-CIC-AndMal-2020" dataset are interpreted and discussed. The results obtained in the dataset are presented in Table 3. According to the test results, the XGBoost algorithm showed the most successful performance among the classifiers used without hyperparameter optimization with the AGA, with an accuracy rate of 99.82%. The second-best performance was the RF algorithm with an accuracy rate of 98.72%. Looking at Table 3, the SVM obtained the most unsuccessful results. The accuracy rate of the SVM algorithm was found to be 73.07%. The NB, KNN, and LR algorithms have obtained similar accuracy values in the range of 89–90%. The GB, ANN, and DNN achieved accuracy values of over 90%. The accuracy values of the GB, ANN, and DNN algorithms are 96.32%, 91.17%, and 93.82%, respectively. As a result of the hyperparameter optimization performed using the AGA technique, the XGB-AGA showed the most successful result. The XGB-AGA showed the most successful performance of all the experiments with its 99.82% accuracy rate performance. At the same time, the results obtained with the RF-AGA and NB-AGA algorithms show that the use of the AGA increases the performance positively.

Table 3. Results of all algorithms.

Algorithms	Accuracy	Recall	Precision	F1-Score
NB	0.891	0.891	0.8942	0.891
RF	0.9742	0.9742	0.9743	0.9742
KNN	0.8982	0.8982	0.903	0.8979
LR	0.8789	0.8789	0.8806	0.8787
SVM	0.7307	0.7307	0.7486	0.7257
GB	0.9632	0.9632	0.9632	0.9632
ANN	0.9117	0.9117	0.9117	0.9117
XGBoost	0.9735	0.9735	0.9736	0.9735
DNN	0.9382	0.9382	0.9382	0.9382
NB-AGA	0.9286	0.9283	0.9328	0.9289
RF -AGA	0.9872	0.9864	0.9891	0.9872
XGB-AGA	0.9982	0.9978	0.9985	0.9982

The graphical representation of the accuracy, recall, precision, and F-score values of the classifiers used in the experiments is shown in Figure 7.

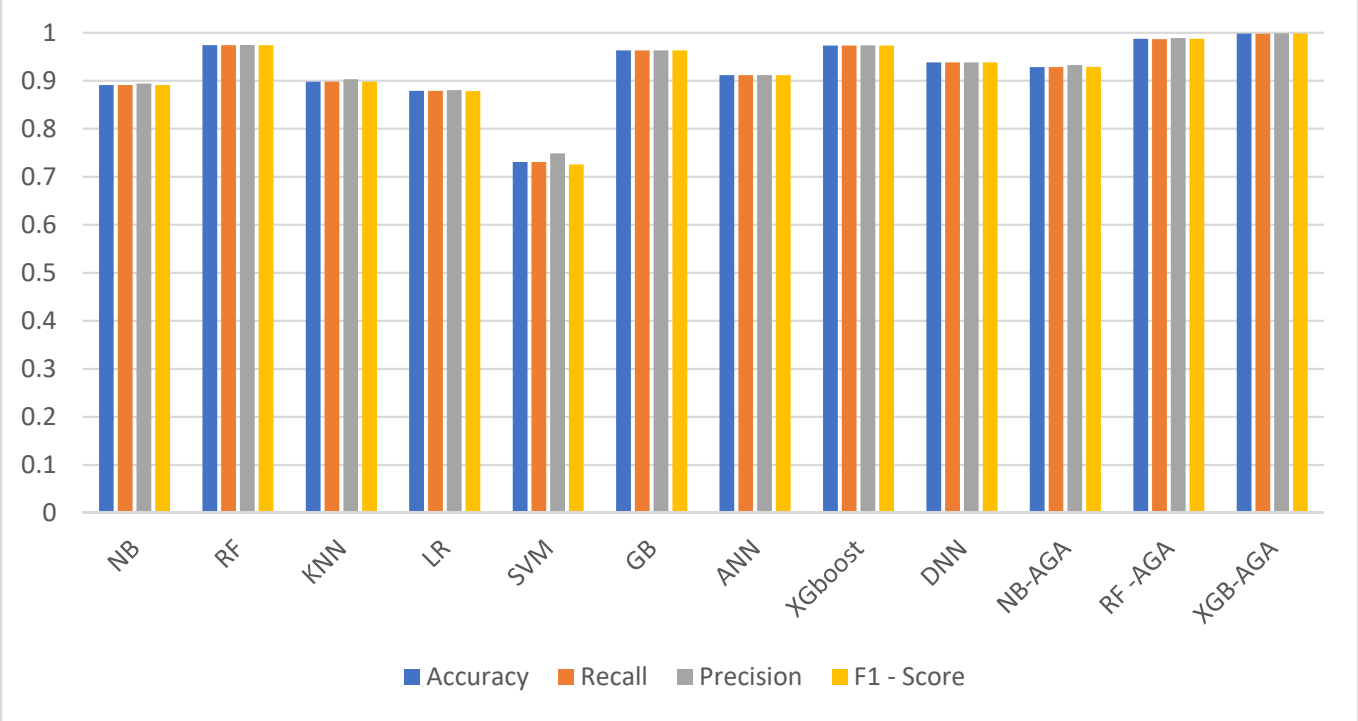


Figure 7. Results of all algorithms.

Confusion matrices showing the estimates obtained on the test set are shown in Figure 8 for each classifier. The "0" index of the confusion matrix indicates benign samples, and the "1" index indicates malware samples. In binary classification problems, the estimation of both classes is important. However, the "FN" parameter, which indicates the number of samples that the classifier estimates as benign but is actually malware, is of vital importance in the safety of autonomous vehicles. The XGB-AGA classifier made only 6 false detections for 2800 test samples. The "FP" number of the XGB-AGA classifier is 4. It is seen that the "FN" and "FP" numbers of the NB, KNN, SVM, and LR algorithms are high and these classifiers make a high amount of incorrect predictions. Although the "FP" number of the GB algorithm is higher than the RF, the "FN" number is lower than the RF. It is also noteworthy that the "FN" number of the XGBoost classifier without using the AGA is lower than that of the RF-AGA classifier. The "FN" and "FP" predictions of the DNN classifier, which is different from the classical machine learning methods, are almost equal and this shows that it makes similarly accurate predictions for both classes.

The ROC curve shows the ability of the classifier to discriminate between the two classes. The area under the curve shows the AUC score. The larger this area, the more developed the classifier's discrimination ability. Figure 9 shows the ROC curve and AUC scores of each classifier used in the experiments. Looking at Figure 9, it is seen that the XGB-AGA classifier exhibits the most successful discrimination ability with an AUC score of 0.99. Secondly, the RF-AGA classifier obtained an AUC score of 0.98. The RF and XGBoost obtained an AUC score of 0.97, the GB 0.96, and the DNN 0.94. The KNN, LR, NB, and SVM classifiers performed at 0.90 and below.

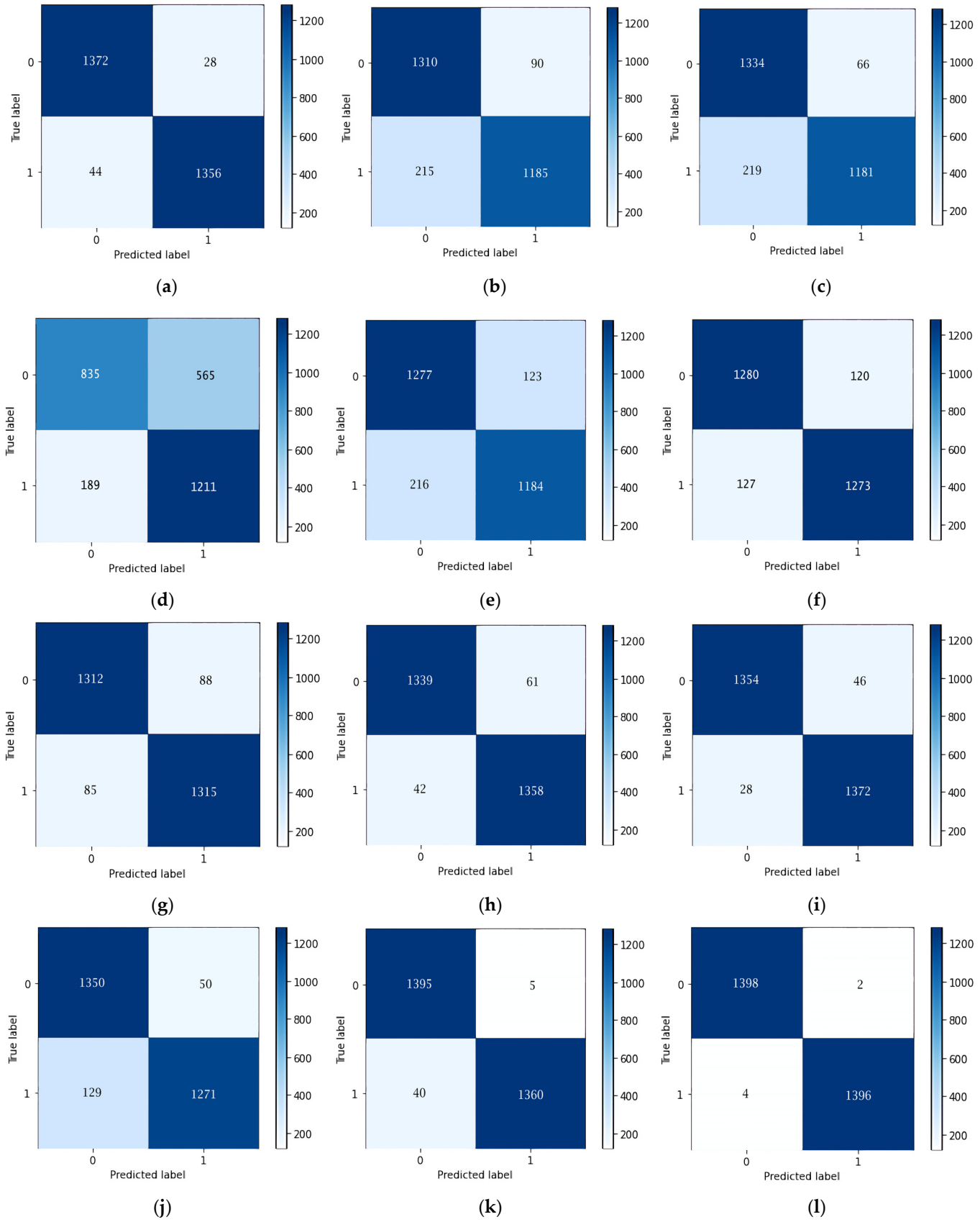


Figure 8. Confusion matrix for all algorithms: (**a**) RF, (**b**) NB, (**c**) KNN, (**d**) SVM, (**e**) LR, (**f**) ANN, (**g**) DNN, (**h**) GBoost, (**i**) XGBoost, (**j**) NB-AGA, (**k**) RF-AGA, (**l**) XGB-AGA.

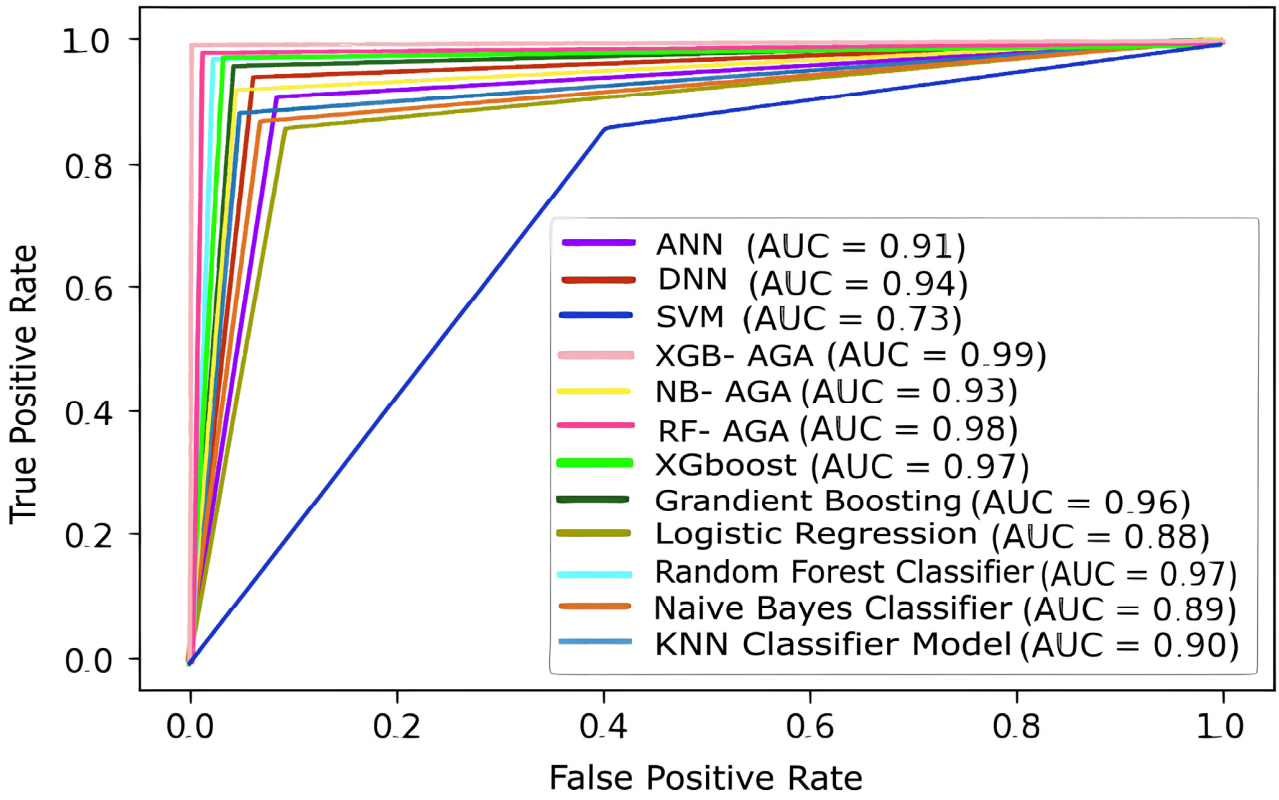


Figure 9. ROC curve for all algorithms.

Three different analysis approaches, static, dynamic, and hybrid, are used for Android malware detection. In addition to the literature, in Table 4, a comparison of the current static, dynamic, and hybrid analysis studies with the proposed approach is presented. The most widely used method for Android malware detection is static analysis. The biggest advantage of static analysis is that early measures are taken by analyzing the application before it is run. In current studies using static analysis, Zhang et al. [11] and Arslan [36] used the deep learning approach. Deep learning methods have high accuracy, but deep learning models consume large amounts of resources and have high computational costs [37,38]. Despite this, accuracies of 0.9666 and 0.9818 were achieved, respectively, in these studies. Milosevic et al. [11], Şahin et al. [39], and Mat et al. [40] classified static analysis information using classical machine learning methods. In these studies, accuracy values of 0.9510, 0.9516, and 0.9110 were obtained, respectively. Atacak et al. [41] used a CNN for feature extraction and ANFIS for classification. As a result of the study, they reached 0.9466 accuracy. In studies using ensemble learning-based static analysis, Atacak [42] achieved an accuracy of 0.9933 with a fuzzy logic and machine learning-based stacking method, and Xie [43] achieved a 0.9866 accuracy with a genetic algorithm and machine learning-based stacking method. The results obtained using static analysis are high. However, the results of studies using machine learning are lower than our study. Although the results of the studies using the ensemble learning approach are similar to our study, they are lower than the results we obtained.

The dynamic analysis approach takes time to implement compared to static analysis, but it is difficult for applications to bypass detection systems due to their behavior at runtime. Park and Choi [9] classified the features obtained using dynamic analysis with random forest. As a result of the experiment, an accuracy of 0.9290 was obtained. Xiao et al. [44] reached a 0.9370 accuracy using dynamic analysis and LSTM architecture. Islam et al. [45] found an accuracy value of 0.95 with the weighted voting method. These studies, in which dynamic analysis was performed, also obtained lower results than our proposed approach.

Table 4. Comparison of results.

Authors	Analysis Approach	Classification Techniques	Accuracy
Zhang et al. [10]	Static	CNN	0.9660
Milosevic et al. [11]	Static	SVM	0.9510
Atacak et al. [41]	Static	CNN + ANFIS	0.9466
Arslan [36]	Static	DNN	0.9816
Şahin et al. [39]	Static	Linear Regression	0.9516
Mat et al. [40]	Static	Naïve Bayes	0.9110
Atacak [42]	Static	FL-BDE	0.9933
Xie et al. [43]	Static	GA-StackingMD	0.9866
Park and Choi [9]	Dynamic	Random Forest	0.9290
Xiao et al. [44]	Dynamic	LSTM	0.9370
Islam et al. [45]	Dynamic	Weighted Voting (RF, KNN, MLP, DT, SVM, LR)	0.95
Rodrigo et al. [12]	Hybrid	DNN	0.9110
Our approach	Hybrid	XGB-AGA RF-AGA	0.9985 0.9872

Using hybrid analysis approaches, Rodrigo et al. [12] achieved an accuracy of 0.9110 using deep neural networks. In our study, PSO-based feature selection was performed using hybrid analysis. At the same time, the performance of machine learning algorithms is optimized with the AGA. Our presented detection approach, XGB-AGA and RF-AGA, has achieved successful detection accuracy compared with existing studies.

7. Conclusions

The safety concerns of self-driving vehicles have increased with the increase in communications that could cause significant damage to vehicles. The damage occurs when any external device is connected, and this device is infected with any malicious code in the vehicles, and this connection happened through any external communication path. The malicious code or malicious program can enter the internal network of the vehicles and cause damage to them. In an integrated automotive environment, where answers must be evaluated in real time, high accuracy and speed are essential to identify harmful activities and prevent damage to vehicles. In this study, a machine learning-based detection approach using hybrid analysis-based particle swarm optimization and an adaptive genetic algorithm is presented for Android malware detection in auto-driving vehicles. The test of the proposed approach was carried out on the "CCCS-CIC-AndMal-2020" data. The hybrid features in the dataset were selected with the PSO algorithm and classified by machine learning algorithms. Random forest and XGBoost algorithms obtained the most successful classification results in the experiments. An adaptive genetic algorithm has been used for the hyperparameter optimization of these algorithms and the current accuracy rates have been increased. In the experiments, the XGBoost algorithm showed the best performance using the AGA with an accuracy rate of 99.82%. The random forest algorithm reached a 98.72% accuracy rate with the AGA. AGA-based classifiers proposed in this study were compared with different classifiers such as NB, KNN, SVM, LR, GB, ANN, and DNN. As a result, it has been observed that feature selection using PSO and hyperparameter optimization using an AGA positively improves performance in Android malware detection. At the same time, it was seen that the results obtained in this study were more successful than similar studies in the literature.

In future studies, testing with more applications can be performed to protect autonomous vehicles from Android malware. In addition, with low computational cost

classification models developed based on a 1D convolutional neural network, the performance can be increased without using PSO and AGA methods.

Author Contributions: The first author (L.H.) conducted experiments and wrote the manuscript, software. The second author (İ.A.D.) was responsible for supervision, project administration, and correcting the direction of the work. The third author (K.K.) revised the manuscript, validation. All authors have read and agreed to the published version of the manuscript.

Funding: The authors received no financial support for the research, authorship, or publication of this paper.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We would like to thank Gazi University Mobile Security Laboratory (MobSecLab) and researchers for their valuable contributions to the experiments and technical support required for our study.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ackerman, M.; Ben-David, S. Measures of clustering quality: Aworking set of axioms for clustering. In Proceedings of the NIPS'08: Proceedings of the 21st International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–10 December 2008; pp. 121–128.
2. Cui, J.; Liew, L.S.; Sabaliauskaite, G.; Zhou, F. A review on safety failures, security attacks, and available countermeasures for autonomous vehicles. *Ad Hoc Netw.* **2019**, *90*, 101823. [CrossRef]
3. *C. S. I. Technical Report*; Computer Security Institute: Orlando, FL, USA, 2007.
4. Doğru, İ.A. *Mobile Security Laboratory*; Department of Computer Engineering, Gazi University—Faculty of Technology: Ankara, Turkey, 2017. Available online: https://mobseclab.gazi.edu.tr/ (accessed on 1 December 2022).
5. Symantec Corp. *Symantec Global Internet Security Threat Report*; Symantec Corp: San Francisco, CA, USA, 2008. Available online: http://www.symantec.com/ (accessed on 9 April 2023).
6. Riggs, C.; Rigaud, C.-E.; Beard, R.; Douglas, T.; Elish, K. A Survey on Connected Vehicles Vulnerabilities and Countermeasures. *J. Traffic Logist. Eng.* **2018**, *6*, 11–16. [CrossRef]
7. Hoppe, T.; Kiltz, S.; Dittmann, J. Applying Intrusion Detection to Automotive IT—Early Insights and Remaining Challenges. *J. Inf. Assur. Secur.* **2009**, *4*, 226–235. Available online: http://www.car-2-car.org/ (accessed on 1 December 2022).
8. Makandar, A.; Patrot, A. Malware Image Analysis and Classification using Support Vector Machine. *Int. J. Adv. Trends Comput. Sci. Eng.* **2015**, *4*, 1–5.
9. Park, S.; Choi, J.Y. Malware Detection in Self-Driving Vehicles Using Machine Learning Algorithms. *J. Adv. Transp.* **2020**, *2020*, 3035741. [CrossRef]
10. Zhang, N.; Tan, Y.-A.; Yang, C.; Li, Y. Deep learning feature exploration for Android malware detection. *Appl. Soft Comput.* **2021**, *102*, 107069. [CrossRef]
11. Milosevic, N.; Dehghantanha, A.; Choo, K.-K.R. Machine learning aided Android malware classification. *Comput. Electr. Eng.* **2017**, *61*, 266–274. [CrossRef]
12. Rodrigo, C.; Pierre, S.; Beaubrun, R.; El Khoury, F. BrainShield: A Hybrid Machine Learning-Based Malware Detection Model for Android Devices. *Electronics* **2021**, *10*, 2948. [CrossRef]
13. Thomas, Z.; Abdelwahed, S. Active malware countermeasure approach for mission critical systems. In Proceedings of the 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Orlando, FL, USA, 6–10 November 2017; pp. 632–638.
14. Iqbal, S.; Haque, A.; Zulkernine, M. Towards a security architecture for protecting connected vehicles from malware. In Proceedings of the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Kuala Lumpur, Malaysia, 28 April–1 May 2019; pp. 1–5.
15. Arp, D.; Spreitzenbarth, M.; Hübner, M.; Gascon, H.; Rieck, K. Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. In Proceedings of the NDSS'14, San Diego, CA, USA, 23–26 February 2014. [CrossRef]
16. Kumar, R.; Zhang, X.; Wang, W.Y.; Khan, R.U.; Kumar, J.; Sharif, A. A Multimodal Malware Detection Technique for Android IoT Devices Using Various Features. *IEEE Access* **2019**, *7*, 64411–64430. [CrossRef]
17. Jeon, S.; Moon, J. Malware-Detection Method with a Convolutional Recurrent Neural Network Using Opcode Sequences. *Inf. Sci.* **2020**, *535*, 1–15. [CrossRef]

18. Abdi, A. Three types of Machine Learning Algorithms List of Common Machine Learning Algorithms. *ResearchGate* **2016**, 1–27. [CrossRef]
19. Keyes, D.S.; Li, B.; Kaur, G.; Lashkari, A.H.; Gagnon, F.; Massicotte, F. EntropLyzer: Android Malware Classification and Characterization Using Entropy Analysis of Dynamic Characteristics. In Proceedings of the 2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge (RDAAPS), Hamilton, ON, Canada, 18–19 May 2021. [CrossRef]
20. Canadian Institute for Cybersecurity (CIC). *CCCS-CIC-AndMal-2020. Canadian Institute for Cybersecurity (CIC) Project in Collaboration with Canadian Centre for Cyber Security (CCCS)*; Canadian Institute for Cybersecurity (CIC): Fredericton, NB, Canada, 2020. Available online: https://www.unb.ca/cic/datasets/andmal2020.html (accessed on 9 April 2023).
21. Pisner, D.A.; Schnyer, D.M. Support Vector Machine. In *Machine Learning: Methods and Applications to Brain Disorders*; Elsevier Inc.: Amsterdam, The Netherlands; Department of Psychology, University of Texas at Austin: Austin, TX, USA, 2019. [CrossRef]
22. Boulesteix, A.-L.; Janitza, S.; Kruppa, J.; König, I.R. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *WIREs Data Min. Knowl. Discov.* **2012**, *2*, 493–507. [CrossRef]
23. Güngör, K.N.; Ayhan Erdem, O.; Doğru, İ.A. Tweet and Account Based Spam Detection on Twitter. In *Artificial Intelligence and Applied Mathematics in Engineering Problems*; Springer: Cham, Switzerland, 2020; Volume 43, pp. 898–905.
24. Patel, H.H.; Prajapati, P. Study and Analysis of Decision Tree Based Classification Algorithms. *Int. J. Comput. Sci. Eng.* **2018**, *6*, 74–78. [CrossRef]
25. Khamis, H.S.; Cheruiyot, W.K.; Kimani, S. Application of k-Nearest Neighbour Classification in Medical Data Mining. *Int. J. Inf. Commun. Technol. Res.* **2014**, *4*, 121–128.
26. Agatonovic-Kustrin, S.; Beresford, R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *J. Pharm. Biomed. Anal.* **2000**, *22*, 717–727. [CrossRef]
27. Zhang, Z. Naïve bayes classification in R. *Ann. Transl. Med.* **2016**, *4*, 241. [CrossRef] [PubMed]
28. Friedman, J.H. Greedy Function Approximation—A Gradient Boosting Machine. *Statistics* **2001**, *29*, 1189–1232. [CrossRef]
29. Chen, T.; Guestrin, C. XGBoost: A scalable tree boosting system. In Proceedings of the KDD'16: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794. [CrossRef]
30. del Valle, Y.; Venayagamoorthy, G.K.; Mohagheghi, S.; Hernandez, J.C.; Harley, R.G. Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Trans. Evol. Comput.* **2008**, *12*, 171–195. [CrossRef]
31. Singh, V. Sunflower leaf diseases detection using image segmentation based on particle swarm optimization. *Artif. Intell. Agric.* **2019**, *3*, 62–68. [CrossRef]
32. Ab Wahab, M.N.; Nefti-Meziani, S.; Atyabi, A. A Comprehensive Review of Swarm Optimization Algorithms. *PLoS ONE* **2015**, *10*, e0122827. [CrossRef]
33. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
34. Muniyappan, S.; Rajendran, P. Contrast Enhancement of Medical Images through Adaptive Genetic Algorithm (AGA) over Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). *Multimedia Tools Appl.* **2019**, *78*, 6487–6511. [CrossRef]
35. Sharma, M. Role and Working of Genetic Algorithm in Computer Science. *Int. J. Comput. Appl. Inf. Technol.* **2013**, *II*, 27–32.
36. Arslan, R.S. AndroAnalyzer: Android malicious software detection based on deep learning. *PeerJ Comput. Sci.* **2021**, *7*, e533. [CrossRef]
37. Duman, E.; Erdem, O.A. Anomaly Detection in Videos Using Optical Flow and Convolutional Autoencoder. *IEEE Access* **2019**, *7*, 183914–183923. [CrossRef]
38. Çinarer, K.; Kiliç, G. Diabetic Retinopathy Detection with Deep Transfer Learning Methods. In *Intelligent and Fuzzy Techniques for Emerging Conditions and Digital Transformation*; Springer: Cham, Switzerland, 2022; Volume 2, pp. 147–154.
39. Şahin, D.; Kural, O.E.; Akleylek, S.; Kılıç, E. A novel permission-based Android malware detection system using feature selection based on linear regression. *Neural Comput. Appl.* **2021**, *35*, 4903–4918. [CrossRef]
40. Mat, S.R.T.; Ab Razak, M.F.; Kahar, M.N.M.; Arif, J.M.; Firdaus, A. A Bayesian probability model for Android malware detection. *ICT Express* **2022**, *8*, 424–431. [CrossRef]
41. Atacak, İ.; Kılıç, K.; Doğru, I.A. Android malware detection using hybrid ANFIS architecture with low computational cost convolutional layers. *PeerJ Comput. Sci.* **2022**, *8*, e1092. [CrossRef]
42. Atacak, İ. An Ensemble Approach Based on Fuzzy Logic Using Machine Learning Classifiers for Android Malware Detection. *Appl. Sci.* **2023**, *13*, 1484. [CrossRef]
43. Xie, N.; Qin, Z.; Di, X. GA-StackingMD: Android Malware Detection Method Based on Genetic Algorithm Optimized Stacking. *Appl. Sci.* **2023**, *13*, 2629. [CrossRef]
44. Xiao, X.; Zhang, S.; Mercaldo, F.; Hu, G.; Sangaiah, A.K. Android malware detection based on system call sequences and LSTM. *Multimedia Tools Appl.* **2019**, *78*, 3979–3999. [CrossRef]
45. Islam, R.; Sayed, M.I.; Saha, S.; Hossain, M.J.; Masud, M.A. Android Malware Classification Using Optimum Feature Selection and Ensemble Machine Learning. *Internet Things Cyber-Phys. Syst.* **2023**, *3*, 100–111. [CrossRef]