mathematics

MDPI

Article

Fuzzy Integral-Based Multi-Classifiers Ensemble for Android Malware Classification

Altyeb Taha *, Omar Barukab and Sharaf Malebary

Department of Information Technology, Faculty of Computing and Information Technology in Rabigh, King Abdulaziz University, Jeddah, Saudi Arabia; obarukab@kau.edu.sa (O.B.); smalebary@kau.edu.sa (S.M.)
* Correspondence: aaataha@kau.edu.sa

Abstract: One of the most commonly used operating systems for smartphones is Android. The open-source nature of the Android operating system and the ability to include third-party Android apps from various markets has led to potential threats to user privacy. Malware developers use sophisticated methods that are intentionally designed to bypass the security checks currently used in smartphones. This makes effective detection of Android malware apps a difficult problem and important issue. This paper proposes a novel fuzzy integral-based multi-classifier ensemble to improve the accuracy of Android malware classification. The proposed approach utilizes the Choquet fuzzy integral as an aggregation function for the purpose of combining and integrating the classification results of several classifiers such as XGBoost, Random Forest, Decision Tree, AdaBoost, and LightGBM. Moreover, the proposed approach utilizes an adaptive fuzzy measure to consider the dynamic nature of the data in each classifier and the consistency and coalescence between each possible subset of classifiers. This enables the proposed approach to aggregate the classification results from the multiple classifiers. The experimental results using the dataset, consisting of 9476 Android goodware apps and 5560 malware Android apps, show that the proposed approach for Android malware classification based on the Choquet fuzzy integral technique outperforms the single classifiers and achieves the highest accuracy of 95.08%.

Keywords: Android malware classification; ensemble learning; choquet fuzzy integral

Citation: Taha, A.; Barukab, O.; Malebary, S. Fuzzy Integral-Based Multi-Classifiers Ensemble for Android Malware Classification. *Mathematics* **2021**, *9*, 2880. https://doi.org/10.3390/math9222880

Academic Editor: Abeer Alsadoon

Received: 1 October 2021
Accepted: 9 November 2021
Published: 12 November 2021

1. Introduction

The use of smartphones is continuously increasing in many aspects of our daily lives. Users are used to storing a considerable amount of their private information on their smartphones. Smartphones are equipped with many sensors that collect valuable information about the user, such as gestures, locations, video, and audio data. The availability of this important information in the smartphones motivates the malware developers and attackers to penetrate the smartphones and steal the private information of the users. The number of smartphone users has increased rapidly in the past few years. Currently, Android is the most widely used smartphone operating system in the world, dominating 70.69% of the market share as of April 2020. [1].

Recent studies suggest that the number of Android apps has increased in recent years due to the open-source nature of Android and the popularity of Android-based smartphones. Moreover, Android-based smartphones have become a widely used control system for the Internet of Things (IoT) [1]. Based on studies by Statista [2], the number of Android apps has grown rapidly since December 2009, and the current number of Android apps existing on Google Play Store in 2020 has reached 2.96 million apps after exceeding 1 million apps in July 2013. This increase in the number of Android apps is due to the key features of Android such as open source nature and ease of use. Privacy issues arose that are of concern to both businesses and users. This is because malware developers intend to deceive users with apps that appear to be useful but have unnoticed malicious intentions.

According to GDATA [3], security experts found more than 4.18 million malicious apps in 2019 and the average number of new Android malware is 11,500 apps per day.

Because Android allows users to download and install programs from Google Play, the official store for Android apps, as well as several third-party markets, it has become the most targeted platform by malware software makers. Due to the lack of effective authentication techniques, malware developers can upload their malicious APPs to Google Play and other third-party stores [4], and evade smartphone users by employing various techniques such as repackaging or encryption, dynamic execution, and code obfuscation [5]. Finance charges, changes to user settings, data theft, privilege escalation, and remote control are all examples of damage caused by malicious Android apps. Currently, malware app developers are employing sophisticated ways to create extremely complicated programs that may easily escape detection by the most advanced anti-malware software [6].

Machine learning approaches have been applied successfully in the field of security [7]. Researchers offer many techniques based on machine learning algorithms to identify Android malware apps, which are divided into two categories: static and dynamic analysis [8]. Static analysis can identify malware by extracting significant features from the APK without executing it and using these features to differentiate between malware and goodware applications. In [9–11], the authors used static analysis to obtain the app's key features, such as the required permissions. Compared with dynamic analysis, static analysis can detect malware quickly and prevent it from being installed. The identification of more complex Android malware apps that employ polymorphism, code obfuscation, and repackage methods is a problem for static analysis. Dynamic analysis is based on installing and executing Android applications in an environment and tracing their execution or behavior [12]. DroidDolphin [13], Crowdroid [14], and DroidWard [15] use dynamic analysis to screen runtime activities such as system calls and API calls and utilize supervised learning algorithms to distinguish between malware and goodware apps.

Because the success of a single classifier is dependent to some extent on the Android app's unique characteristics, it has inherent individual uncertainty and limitations. When compared with a single classifier, the combination of classifiers has the benefit of minimizing the variation of estimated error and improving classification performance. Ensemble learning is a machine learning approach that uses many learning algorithms rather than a single algorithm to generate a single optimal prediction model; it has shown to be an effective solution in a variety of disciplines. Ensemble learning approaches have been shown empirically and theoretically to outperform single weak methods, particularly when dealing with complicated, high-dimensional prediction problems [16]. Bagging [17], Boosting [18], and Stacking [19] are the most often used ensemble learning methods. Stacking is a fusion approach that combines many machine learning algorithms (base models) structured in at least one layer and then employs another machine learning algorithm (meta model) to achieve higher classification performance than any of its basic algorithms [20]. In [21,22], the authors collected a large number of significant features and strings from Android APK files and used these features to create multiple ensemble machine learning techniques to identify Android malware apps.

There are intrinsic interactions between several basic classifiers; these interactions might be positive synergy in nature, in which case the basic classifiers collaborate and strengthen one another. The ensemble system may use the strengths of the different classifiers and overcome their weaknesses to attain a better level of accuracy than any single classifier. The fuzzy integral has the advantages of characterizing the significance of basic classifiers by fuzzy measure and modeling the coalitions and interactions among every possible coalition (subset) of classifiers. This study presents a multi-classifier ensemble based on fuzzy integrals to enhance the accuracy of Android malware classification. The proposed method employs Choquet fuzzy integral as an aggregation function to fuse the prediction results of multiple classifiers, including XGBoost, Random forest, Decision tree, AdaBoost, and LightGBM, by taking into account both the significance of each classifier as well as the consistency and coalition among each possible subset of classifiers based

on adaptive fuzzy measures. The proposed approach is able to aggregate more meaningful prediction results from multiple classifiers using the adaptive fuzzy measures. The following are the primary contributions of the suggested approach:

- Proposes a novel fuzzy integral based multi-classifiers ensemble for Android malware classification. The proposed approach has the capability of aggregating the classification results from multiple classifiers by taking into account both the significance of each classifier as well as the consistency and coalition among each possible subset of classifiers based on fuzzy measures.
- For successful detection of Android malware apps, the proposed approach presents an adaptive fuzzy measure based on dynamic data in single classifiers and consistency and coalition among each potential subset of classifiers.
- The proposed approach's performance was evaluated through a series of experiments, and the results indicate that it outperforms both individual classifiers and other approaches.

The rest of the paper is organized as follows. Section 2 presents the related work on Android malware detection. Section 3 presents the fuzzy measures and Choquet integral. Section 4 explains the proposed approach for Android malware classification. Results and discussion are illustrated in Section 5. Section 6 concludes this paper.

2. Related Work

With the number of Android malware apps growing fast, several researchers have proposed approaches based on machine learning techniques for the classification of Android malware apps.

Arp et al. [22] developed an app called "Drebin" for detecting Android malware apps. They used the API calls, Android permissions, and network addresses as features to differentiate between goodware and malware apps. Their experimental results were based on a dataset consisting of 5560 malware apps and 123,453 goodware apps from numerous app markets. Drebin achieved classification accuracy of 94% and a false positive rate of 1%. Firdaus et al. [23] presented an approach for Android malware apps detection based on features investigation. The features include Android permissions and directory paths. They used three machine learning algorithms: voted perceptron (VP), radial basis function network (RBFN), and multilayer perceptron (MLP) in the conducted experiments based on Drebin and AndroZoo [24] datasets. Their proposed approach based on MLP achieved detection accuracy of 90% and true positive rate of 87%. Hu et al. [25] presented an approach using an ensemble learning classifier with the feature selection algorithm, a sliding window is utilized inside the ensemble classifier. Zhang et al. [26] presented a new approach using the Markov blanket for feature selection, SVM is employed for classification of Android malware apps, and Drebin dataset is used in their experimental results.

Coronado-De-Alba et al. [27] suggested an ensemble-learning approach for Android malware detection. They used a dataset consisting of 1531 malware and 1531 goodware apps; the malware apps were selected from the Drebin dataset [22], and the goodware samples were obtained from the Google Play Store. They used two features selection algorithms, RELIEF and Chi-squared, and the Random Forest algorithm for the detection of Android malware. Peiravian and Zhu [28] introduced a machine-learning-based model for Android malware detection. They used Android app permission and API calls as main features to distinguish between goodware and malware apps.

Wang et al. [29] utilized Decision Tree, Linear SVM, Logistic regression, and Random forest for malware apps detection-based static analysis. They used the apps static features for machine learning training. Based on the experimental results, their proposed approach detects malware apps with false positive rate of 0.06% using the Logistic Regression algorithm based on a dataset containing 217,619 goodware and 18,363 malware apps. Talha et al. [30] suggested a classification approach based on permissions "APK Auditor" to categorize the Android apps as goodware or malware which achieved 88% accuracy with

a specificity of 0.925. They used a dataset consisting of 8762 apps including 1853 goodware apps and 6909 malware apps.

DroidDolphin [13] is a dynamic analysis approach that uses machine learning to classify Android malware. It analyzes Android apps based on obtaining the apps details from API calls and activities by running the apps on virtual environments and attained a precision of 86.1% using the SVM algorithm. Milosevic et al. [31] proposed an Android malware classifier based on the ensemble learning model. They used Android permissions and source-code-based features. The ensemble classifier used contained linear regression, Decision Trees, SVM, C.45, Random Tree, and Random Forests algorithms.

In our previous research [32], we proposed a hybrid approach for the classification of Android malware by integrating the fuzzy C-means clustering (FCM) algorithm with the light gradient boosting machine (LightGBM). FCM is used to cluster the Android permissions, while LightGBM is utilized to classify the Android apps based on the app's permissions and their clusters resulting from FCM. Using a widely used dataset consisting of benign and malware Android apps, the experimental results show that the suggested approach attained an accuracy of 94.63%.

In our previous research [9], we suggested an evolving hybrid neuro-fuzzy classifier (EHNFC) for Android malware classification using permission-based features. We adapted the evolving clustering algorithm to incorporate an adaptive method for updating the radii and centers of clustered permission-based features. The results show that the proposed method classifies Android malware with an accuracy of 90%. In our previous research [10], we introduced an approach for Android malware classification using the adaptive neuro fuzzy inference systems (ANFIS), an information gain method utilized to choose the important Android permissions. The proposed method attained an accuracy of 75%.

In our previous research [11], we introduced an adaptive neuro-fuzzy inference system with fuzzy c-means clustering (FCM-ANFIS) for Android malware classification. The experimental results show that a classification accuracy of 91% was achieved by the proposed method. Table 1 summarizes the discussed related works.

Table 1. Summary of the discussed related works.

Approach	Description of Method	Accuracy	Limitations
Altaher [9]	Android permissions and adaptive neuro fuzzy inference systems (ANFIS)	75%	Static analysis method and lacks the dynamic real-time inspection
DroidDolphin [13]	API calls and activities by running the apps on virtual environments and SVM algorithm	86.1%	Dynamic analysis consuming resources and takes a long time
Arp et al. [22]	Used API calls, Android permissions and network addresses as features and support vector machine (SVM) algorithm	94%	Exhibits the inherent limitations of static analysis
Firdaus et al. [23]	Android permissions used as features, and three machine learning algorithms: voted perceptron (VP), radial basis function net-work (RBFN), and multilayer perceptron (MLP)	90%	Static analysis method and lacks the dynamic real-time inspection
Hu et al. [25]	Android permissions, application actions, and API were utilized as features An ensemble learning classifier was used as classifier	96%	Unable to detect new Android malware apps
Talha et al. [30]	Android permissions and machine learning algorithms	88%	Static analysis method and lacks the dynamic real-time inspection

Table 1. *Cont.*

Approach	Description of Method	Accuracy	Limitations
Taha and Malebary [32]	Android permissions and fuzzy C-means clustering (FCM) algorithm with the light gradient boosting machine (LightGBM)	94.63%	Exhibits the inherent limitations of static analysis
Awan et al. [33]	Spatial attention and convolutional neural network (SACNN) based on deep learning framework	97.42%	Lack of exploration in the data augmentation and the feature engineering domains
Hemalatha et al. [34]	Visualization-based method, where malware binaries are depicted as two-dimensional images and classified by a deep learning model	98.23%	Consumes re-sources and takes long time for training
Nisa et al. [35]	The features that are extracted from malware images are then classified using different variants of support vector machine (SVM), k-nearest neighbor (KNN), decision tree (DT), and other classifiers	99.3%	Used a pre-trained model and cannot detect new Android malware apps

3. Choquet Fuzzy Integral

The idea of the Choquet integral was initially presented in capacity theory [36]. It is utilized as a fuzzy integral by considering a fuzzy measure [37]. Then, the Choquet integral is rediscovered in [38,39] by Murofushi and Sugeno. Choquet integral aims to integrate functions based on the fuzzy measures. For Android malware classification, the fuzzy measures based on the Choquet integral were utilized to state a weight for each set of classifiers. Since classifiers can be dependent, therefore, it is possible to enable modeling of the interaction existing between classifiers. The description of the Choquet integral and fuzzy measures is shown in [38] where: $C = \{c_1, c_2, \ldots, c_N\}$ is a set of N machine learning algorithms, and P(C) represents the performance of C or set of all subsets of C.

Definition 1. *Suppose C is a finite set of machine learning algorithms, the discrete fuzzy measure on C is the mapping:*

P(C)→[0, 1], adhering to the requirements below:

(1) $\boldsymbol{\mu}$ *(Ø) = 0, l(C) = 1 (requirements limits).*
(2) *If A, B ∈ P(C) and A ⊂ Z then* $\boldsymbol{\mu}$ *(A) ≤* $\boldsymbol{\mu}$ *(B) (monotonicity).*

where $\boldsymbol{\mu}$ *(S) represents the score of the importance of classifier set S. Therefore, weights on any classifiers combination are also considered as well as the standard weights of each classifier taken independently.*

Definition 2. *Suppose μ is the fuzzy measure on C = 1; 2; . . . , n. The Choquet fuzzy integral of the function mapping:*

f: C→R with relation to $\boldsymbol{\mu}$ *is explained by:*

$$CH_{\mu}(f) = \sum_{i=1}^{n} f_{(i)} \left[\mu(A_{(i)} - \mu\left(A_{(i+1)}\right) \right], \tag{1}$$

where (i) denotes a replacement on C thus

$$f_{(1)} \leq f_{(2)} \leq \ldots \leq f_{(n)}. \text{ Moreover } A_{(i)} = i, \cdots, n, A(n+1) = \varnothing.$$

Because the independency of one classifier from another is not necessary in the fuzzy integral model, it can be utilized in many non-linear situations. The overall evaluation of all possible solutions can be obtained by the fuzzy integral of f with respect to $\boldsymbol{\mu}$. In the

case of independent classifiers, the fuzzy measure is additive, and the Choquet integral matches with the weighted arithmetic mean approach. Thus,

$$CH_{\mu}(f) = \sum_{i=1}^{n} f_i \mu(\{i\}) \tag{2}$$

A Choquet integral is an approach of collection that takes into account both the significance of a classifier and its relations with various classifiers [40]. Suppose G, M $\subset$ X and G $\subset$ M = Ø, g are known as a λ-fuzzy measure when the following condition is satisfied:

$$g(G \cup S) = g(G) + g(M) + \lambda g(G)g(M),\ \lambda\ \epsilon\ (-1, \infty) \tag{3}$$

For the relations between G and M, if $\lambda > 0$, then the multiplicative effect exists; if $\lambda < 0$, then the substitutive effect exists. $\lambda = 0$ shows that G and M are unrelated [41].

Let g_i represent the fuzzy density based on the Choquet fuzzy integral, $g(A_j)$ is attained by λ and g_j as

$$g(A_j) = g_j + g(A_{j+1}) + \lambda g_j g(A_{j+1}) = \frac{1}{\lambda}\left[\prod_{i=j\cdots n} (1 + \lambda g_i) - 1\right] \tag{4}$$

λ is computed by solving the following equation:

$$F(\lambda) = \prod_{j=1}^{n}(1 + \lambda\mu_j) - \lambda - 1 = 0 \tag{5}$$

In [42], it was proved that for a fixed set of μj, $0 < \mu j < 1$, there exists a unique root of $\lambda > -1$, and $\lambda \neq 0$, using Equation (5). In addition, from Equation (5) it can also be seen that if the values of μj are known, then λ can be calculated.

4. The Proposed Fuzzy Integral-Based Multi-Classifiers Ensemble for Android Malware Classification

This research proposes a fuzzy integral-based multi-classifiers ensemble for Android malware classification. The suggested method combines five different basic classifiers, namely XGBoost, Random Forest, Decision Tree, AdaBoost, and LightGBM, with one fusion module based on the Choquet fuzzy integral. The Android app permissions are used as inputs to the five basic classifiers, and the prediction results from all basic classifiers are combined using the Choquet fuzzy integral to determine the final classification results of the Android app. Figure 1 illustrates the proposed approach.

4.1. Dataset and Features Selection

In this study, a dataset containing 9476 Android goodware apps and 5560 malware Android apps was utilized [22]. The Android malware apps in the dataset are from 49 different malware families; examples for the malware families in the dataset included FakeInstaller, DroidKungFu, Goldmaster, and GingerMaster. The dataset is preprocessed and partitioned into training and testing datasets in preparation for evaluation. The dataset includes Android permissions as distinguishing features between goodware and malware applications. The important Android permissions are then carefully determined utilizing the information gain approach to improve classification accuracy. Using essential features that describe the unique dataset characteristics, feature selection algorithms can help improve classification accuracy. As a result, feature selection approaches reduce dataset dimensions by removing Android permissions that aren't needed for the classification process. Redundant or different characteristics can lead to a variety of issues, including a reduction in the efficacy of machine learning algorithms. To achieve an accurate identification of Android malware apps, it is necessary to use an appropriate feature selection method.

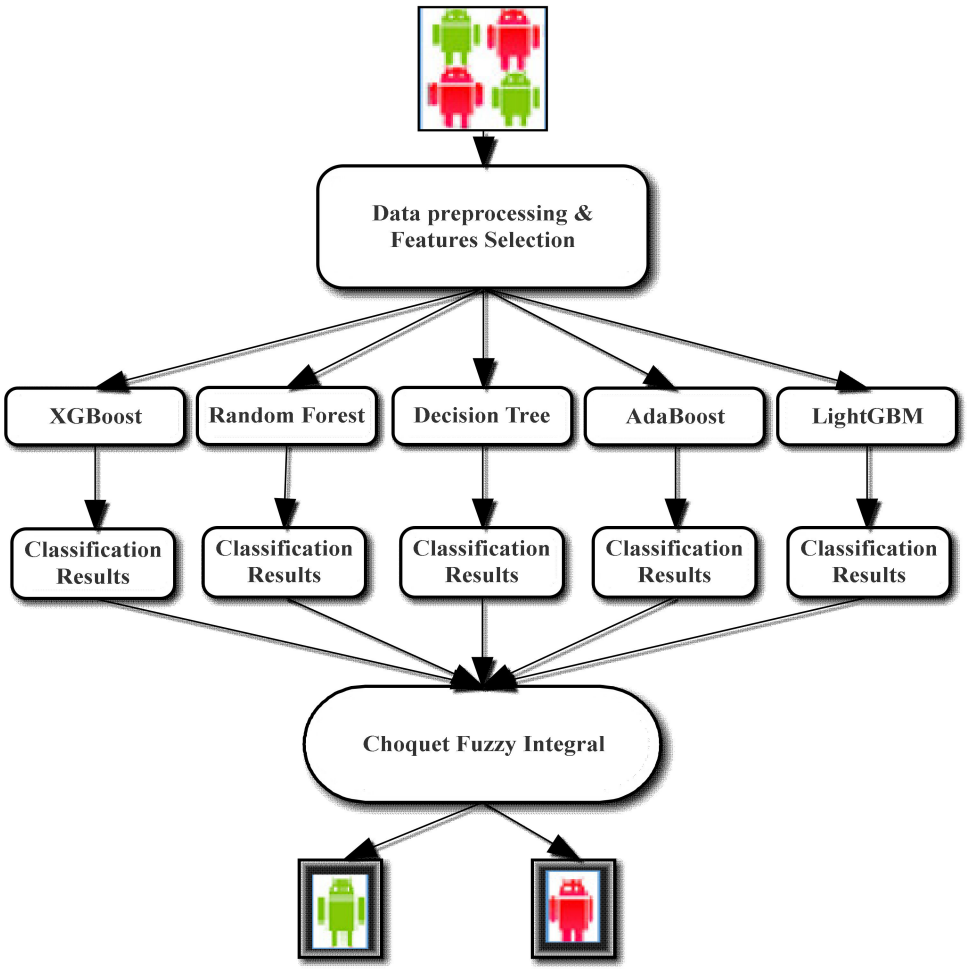

Figure 1. The proposed approach for Android malware classification.

The information gain ratio (IGR) technique [43] is applied to select the essential features for Android malware classification. The IGR technique depends on finding the relationships between the features of an Android app and then calculating and scoring each feature individually. The information gain is utilized to select the important Android apps' permissions in the suggested approach because of its high efficiency and fast computing. Based on the class (malware or goodware), it assesses the best Android permissions. Figure 2 displays examples for the features employed in the proposed model and their information gain ranks. The information gain ranks for the chosen permissions are reflected in the horizontal axis.

4.2. Single Classifiers

Motivated by our previous research on Android malware classification [9,11,32], the XGBoost, Random forest, Decision tree, AdaBoost, and LightGBM algorithms are employed as single classifiers in this study for the classification of Android malware applications. Then, the classification outputs resulting from these classifiers are combined using the Choquet fuzzy integral.

4.2.1. The eXtreme Gradient Boosting (XGBoost)

The XGBoost algorithm suggested by Dr. Chen Tianqi [44] provides good learning via additive training approaches which fuse all expectations of a set of "Ineffective" learners to reduce the variance by adding regularization terms. It has the capabilities of high accuracy, fast running speed, and low computational complexity regardless of whether the

data size is big or small. This method has been successfully applied in classification and prediction approaches.

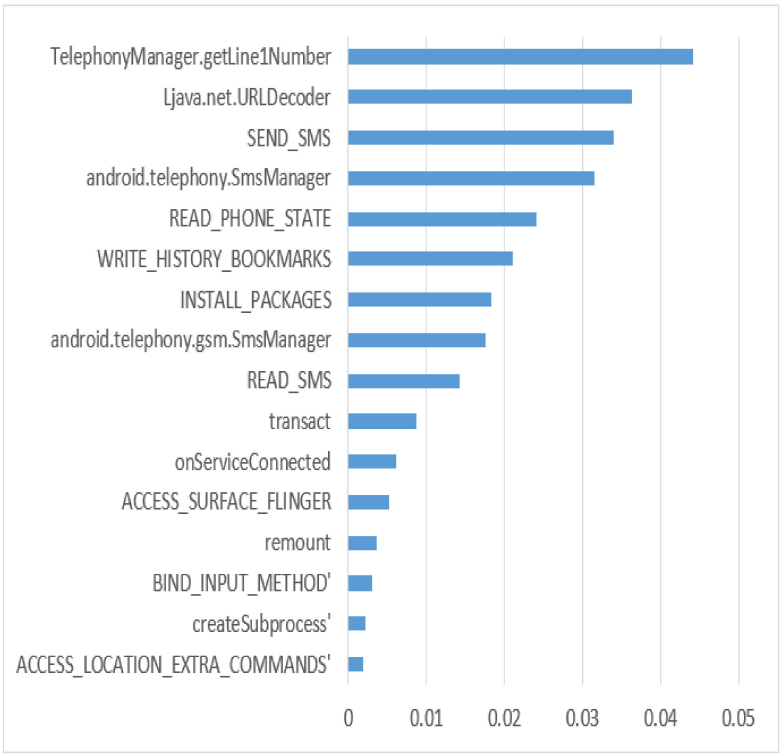

Figure 2. The selected features for Android malware classification.

4.2.2. Random Forest (RF)

Random Forest is an ensemble learning technique that has been very successful as a regression and classification technique. Random forest builds several decision trees that will be utilized by the majority vote to classify a new sample. Each decision tree node uses a subset of features chosen randomly from the entire original set of features [45].

4.2.3. Decision Tree (DT)

DT is a widely employed approach for classification as it is similar to human reasoning and it is simple to explain. Using the decision tree, a new sample can clearly be categorized and the basic idea is to consistently match the corresponding characteristics and associated conditions until a leaf node of the class label has been reached [46].

4.2.4. AdaBoost

Freund and Schapire [47] have proposed AdaBoost, which provides a majority vote over many classifiers. AdaBoost has become one of the most important and powerful classification algorithms in many applications due to its low implementation complexity, high performance, and strong generalization capabilities.

4.2.5. LightGBM

LightGBM is a new GBDT (gradient boosting decision tree) algorithm, suggested by Ke and colleagues in 2017, which has been successfully utilized in several types of data mining operations, such as regression and classification [48]. The LightGBM algorithm includes two novel methods: the gradient-based one-side sampling and the exclusive feature bundling.

The performance of classifiers is influenced by used parameters directly. We utilized the grid search to determine the ideal parameters for each classifier in order to obtain good performance. Table 2 summarizes the used parameters for each classifier.

Table 2. The used parameters.

Classifier	Parameters
LightGBM	Objective = regression, metric = rmse, num_leaves = 80, learning_rate = 0.09, bagging_fraction = 0.7, feature_fraction = 0.7, bagging_frequency = 5, bagging_seed = 2018, verbosity = 1
Random Forest	n_estimators = 50 and random_state = 1
XGBoost	random_state = 1 and learning_rate = 0.01
AdaBoost	n_estimators = 100
Decision Tree	max_depth = 5, min_samples_leaf = 4

4.3. Classifiers Fusion Based on Choquet Integral

Any single classifier has benefits and disadvantages. The performance of a single classifier for Android malware classification is to some extent dependent on the sample's attributes, and there is uncertainty for each classifier. Choquet's integral fusion may synthesize the classification outputs of several classifiers and achieve the highest consistency of the outputs between the varied and consistent outputs. Under uncertain conditions, the Choquet integral fusion is an efficient reasoning strategy. There are several Android permissions that might be exploited by both Android malware and goodware apps, making it difficult to distinguish between the two. As a result, the classification result of several classifiers fused using the Choquet integral is more accurate and suitable for Android malware classification.

4.4. Adaptive Fuzzy Measure

As shown in Equation (1), the Choquet fuzzy integral works on the fuzzy measures (g), which are computed based on the level of significance of each classifier or the level of significance of a subset of classifiers. Let $C = \{c_1, c_2, \ldots, c_N\}$ be the set of N classifiers, T= {M,B} be the set of class-types of Android apps, where M denotes the malware apps and B denotes the benign apps. $P(c_i) = \{p(c_i^M), p(c_i^B)\}$ represents the output of classifier c_i where $p(x_i^M)$ represents the classification score assigned by classifier x_i to the class-type malware app and $p(x_i^B)$ represents the classification score assigned by classifier c_i to the class-type benign app. Generally, the importance of a classifier over a dataset is explained using the confusion matrix as shown in Figure 3.

	Positive prediction	**Negative prediction**
Positive class	**True Positive (TP)**	**False Negative (FN)**
Negative class	**False Positive (FP)**	**True Negative (TN)**

Figure 3. Confusion matrix for a two-class problem.

The confusion matrices are beneficial tools to check the errors that the classifiers are making. Suppose CM_k is the confusion matrix of classifier c_k:

$$CM_k = \begin{bmatrix} n_{11}^k & n_{12}^k & \cdots & n_{1N}^k \\ n_{21}^k & n_{22}^k & \cdots & n_{2N}^k \\ \cdots & \cdots & \cdots & \cdots \\ n_{N1}^k & n_{N2}^k & \cdots & n_{NN}^k \end{bmatrix} \quad (6)$$

where i = j, n_{ij}^k indicates the number of Android apps belonging to class type T_M and are correctly classified by the classifier c_k as class T_M. On the other hand, $i \neq j$, n_{ij}^k denotes the number of Android apps with class type T_M but are incorrectly classified as class T_B by classifier c_k.

As the circumstances between the classification results and training cannot be very consistent, the fuzzy measure must be adaptively adjusted to appropriately cope with the classification situation. However, after the classifiers training stage, the generated confusion matrix will be fixed. Consequently, if the Android malware classifier is constructed based on the static prior fuzzy density, then the classification accuracy of the fusion classifier may be decreased. It is logical to fine tune the fuzzy measure based on the adaptive details in the single classifier outputs.

To tune the fuzzy measure, two issues were considered: firstly, the confidence of classification results for each classifier, and secondly, the consistency between the classifications results for each classifier. For the confidence of each single classifier result, we considered the confidence improvement factor that is represented by α, which reduces the importance of the classifier when the confidence of the classifier results is less significant [49]. For the consistency between the classifier results, we considered the consistency improvement factor, which is represented by β. It reduces the significance of the classifier when the consistency of the classifier results is less important.

From the confusion matrix, a number of class-wise measures are obtained such as accuracy, precision, recall, and F1-score. In this study, the performance, *P*, of classifier *i* is evaluated by considering the two classes simultaneously using the accuracy measure:

For a given Android app, the classification performance by classifier *i* is

$$P_i = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

Let $K \subseteq N$ denote a subset of classifiers with Pk being the performance of classifier K for each $I \in \{1,2, \ldots, n\}$. Then μ_i is the mean performance of classifiers $S \subseteq N$ with $|S| = i$

$$\mu_i = \frac{1}{D} \sum_{S \subseteq N, |S| = i} P_S \quad (8)$$

where $D = \begin{pmatrix} n \\ i \end{pmatrix}$ denotes the number of sets in the level i of measure and $\mu_0 = 0$. The confidence improvement factor α and consistency improvement factor β of classifier *ck* for the Android app a_i are represented as follows:

For single classifiers, the confidence improvement factor is

$$\alpha_i = p_i \quad (9)$$

The confidence improvement factor and consistency improvement factor for a sub set of classifiers are calculated using Equations (10) and (11), respectively, in order to consider the diversity between classifiers:

$$\alpha_i = \frac{1}{n} + \frac{\tanh(100 * (P_i - \mu_i))}{2n} \quad (10)$$

$$\beta_i^k = \prod_{i=1}^{k} \gamma^{ki} \tag{11}$$

where the γ^{ki} denote the consistency between the classification results from classifier k and the classification result from classifier i. Assume classifier k divides the Android app a_i as malware app (X) while classifier i classifies the Android app a_i as a goodware app (Y), then the γ^{ki} is calculated as follows:

$$\gamma^{ki} = \begin{cases} 1 - (s_{kX} - S_{iX}) & X \neq Y \\ 1 & X = Y \end{cases} \tag{12}$$

The updated fuzzy measure is computed as follows:

$$g_i = \alpha_i * \beta_i^k \tag{13}$$

To classify the Android apps using the Choquet fuzzy integral, we used steps in Figure 4.

INPUT:
Dataset based on Android malware and goodware apps, N set of classifiers,
K: any combination of classifiers that is a subset of N
OUTPUT:
The class of the Android app (malware or goodware)

Preprocess the dataset and select the important features.
For K $\subseteq$ N
Use the training and testing datasets as inputs to single classifiers to obtain the confusion matrix CM_k and find P_i which will be employed as initialization of the fuzzy density, using Equations (6) and (7), respectively.
End For
For K $\subseteq$ N do
Compute the confidence improvement factor (α_i^k) by Equations (9) and (10), the consistency improvement factor (β_i^k) by Equation (11),
End for
Based on the confidence improvement factor α_i^k and consistency improvement factor β_i^k of classifier f_k for the Android app a_i, adjust the initialization of the fuzzy density using Equation (13).
Use Equations (3) and (4) to calculate the λ -Fuzzy measure.
For K $\subseteq$ N do
Calculate the Choquet integral of g_i for all the Android samples a_i
End for
For each Android app a_i
The class T_i, with the biggest Choquet integral value is selected as a classification result.
End for

Figure 4. Steps of the proposed approach for Android malware classification.

5. Results and Discussion

To investigate the efficiency of the proposed fuzzy Integral-based multi-classifier ensemble for Android malware classification, in this section we conduct a performance comparison between the proposed approach and the five state-of-the-art single classifiers, i.e., Xgboost, Random Forest, Decision Tree, AdaBoost, and LightGBMM using a real-world dataset. Moreover, we compared the performance of the proposed approach with the other research in the literature for Android malware classification.

The Choquet integral is employed with the fuzzy measures obtained from the single and all possible sets of classifiers to find the final classification of the Android app, either as a malware or goodware app. For further illustration, the classification with the arithmetic mean is considered to demonstrate that utilizing an appropriate fuzzy measure combined with the Choquet integral can achieve classifications that are more accurate.

The results of the classification obtained by each of the classifiers (namely, XGBoost (C1), Random Forest (C2), Decision Tree (C3), AdaBoost (C4), and LightGBM (C5), for 10 Android apps randomly selected from the used dataset are shown in Table 3. The true labels for the Android apps are provided in column Y, while column $\hat{y}\mu$ and $\hat{y}$ C represent the obtained output based on the Choquet fuzzy integral and arithmetic mean. Each classification result denotes the probability of each Android app belonging to malware class 1 (p(y = 1 | x), or goodware class p(y = 0 | x) = 1 − p(y = 1 | x)).

Table 3. Classification of Android Apps using individual classifiers, the Choquet fuzzy integral, and arithmetic mean.

Android Apps	XGBoost	Random Forest	Decision Tree	AdaBoost	Light-GBM	Actual	$\hat{y}$C	$\hat{y}\mu$
App1	0.1944	0.0033	0.0032	0.4870	0	0	0	0
App2	0.7831	0.9936	0.9930	0.5072	0.9920	1	1	1
App3	0.2605	0.1132	0.1135	0.4959	0.1135	0	0	0
App4	0	0.2605	0.4984	0.6121	1	1	1	0
App5	0.1944	0.0033	0.0032	0.4870	0.0034	0	0	0
App6	0.6965	0.8627	0.8628	0.5049	0.8606	1	1	1
App7	0.1944	0.0033	0.0032	0.4870	0	0	0	0
App8	0.7831	0.9858	0.9873	0.5133	0.9877	1	1	1
App9	0.7831	0.9858	0.9873	0.5133	0.9877	1	1	1
App10	0.6965	0.8627	0.8628	0.5049	0.8606	1	1	1

To build the fuzzy measures, we iterated through all the probable sets of classifiers and calculated their performance. The performance of a set of classifiers is computed using the arithmetic mean of the classification results for the classifiers in the set. Then, each Android app is classified as malware or goodware app by selecting the class that belongs to the greatest aggregated output. Lastly, the performance of the classifier(s) is provided by comparing the classification results of the sub-ensemble with the actual results. Table 4 shows the performances of the single and all possible sets of classifiers.

Considering the data in Table 4 and Equation (12), we can construct the fuzzy measure for the single classifiers and all the possible sets of classifiers as shown in Figure 5.

After constructing the fuzzy measure as shown in Figure 5, we can classify each Android application as malware or goodware using the Choquet integral. For example, we will consider the process of classifying the fourth Android app in Table 3. The probabilities of App4 belonging to malware class given by each classifier are PC1 (y = 1 | App4) = 0.0000, PC2 (y = 1 | App4) = 0.2605, PC3 (y = 1 | App4) = 0.4984, PC4 (y = 1 | App4) = 0.6121and PC5 (y = 1 | App4) = 1.0000. Since PC3 (y = 1 | App4) < PC1 (y = 1 | App4) < PC2 (y = 1 | App4) < PC4 (y = 1 | App4) < PC5 (y = 1 | App4), and the coefficients of the fuzzy measures that are significant for the classification of App4 are m({C1,C2, C3, C4,C5}), m({C1,C2, C3, C4}), m({C4, C2, C5}), m({C2, C4}), and m({C2}) (see blue path in Figure 2). Thus, the aggregation of scores for malware class is given by:

Cm(0.0000, 0.2605, 0.4984, 0.6121, 1.0000) = 0.0000 * m({ C1,C2, C3, C4,C5})+ (0.2605 − 0.0000)* m({ C1,C2, C3, C4}) + (0.4984- 0.2605)* m({C4, C2, C5}) + (0.612172 − 0.498429)*

m({C2, C4}) + 1.0000* m({C2}) = 0.0000 *(1) + (0.2605 − 0.0000) *(0.8624)+ (0.4984 − 0.2605)*(0.6698)+ (0.612172 − 0.498429) * (0.4268) + (1 − 0.612172) * (0.9495) = 0.8008. Since 0.8008 > 0.5, the proposed approach classifies the fourth Android app as a malware app. The fuzzy measures utilized in the classification of the fourth Android app are illustrated by the blue paths in Figure 5. However, if the arithmetic mean is considered for the classification of the fourth Android app, the result would be (0 + 0.260513 + 0.498429 + 0.612172 + 1)/5 = 0.4742, and since 0.4742 < 0.5, the classification result would be goodware app, which is an incorrect classification result. Noticing that the true class label for the fourth Android app is 1, the proposed approach correctly classified the fourth Android app, due to its ability to consider the interactions among classifiers.

Table 4. Classification performance of single classifier and all possible sets of classifiers.

Sub-Ensemble of Classifiers	Performance Score	Average Performance Score
Decision Tree	0.9477	0.9408
AdaBoost	0.9366	
Random Forest	0.9495	
XGBoost	0.9224	
LightGBM	0.9482	
Decision Tree, Random Forest	0.9503	0.9465
Decision Tree, XGBoost	0.9503	
Decision Tree, AdaBoost	0.9238	
Decision Tree, LightGBM C5	0.9491	
Random Forest, Decision Tree	0.9482	
Random Forest, AdaBoost	0.9493	
Random Forest, LightGBM	0.9491	
Decision Tree, AdaBoost	0.9480	
Decision Tree, LightGBM	0.9484	
AdaBoost, LightGBM	0.9491	
Decision Tree, Random Forest, XGBoost	0.9503	0.9493
Decision Tree, Random Forest, AdaBoost	0.9503	
Decision Tree, Random Forest, LightGBM	0.9491	
Decision Tree, AdaBoost, LightGBM	0.9491	
Random Forest, XGBoost, AdaBoost	0.9482	
Random Forest, XGBoost, LightGBM	0.9488	
Random Forest, AdaBoost, LightGBM	0.9491	
XGBoost, Decision Tree, LightGBM	0.9486	
XGBoost, Decision Tree, AdaBoost	0.9503	
Decision Tree, Random Forest, Decision Tree, AdaBoost	0.9503	0.9495
Decision Tree, Random Forest, XGBoost, LightGBM	0.9488	
Decision Tree, Random Forest, XGBoost, AdaBoost, LightGBM	0.9488	0.9488

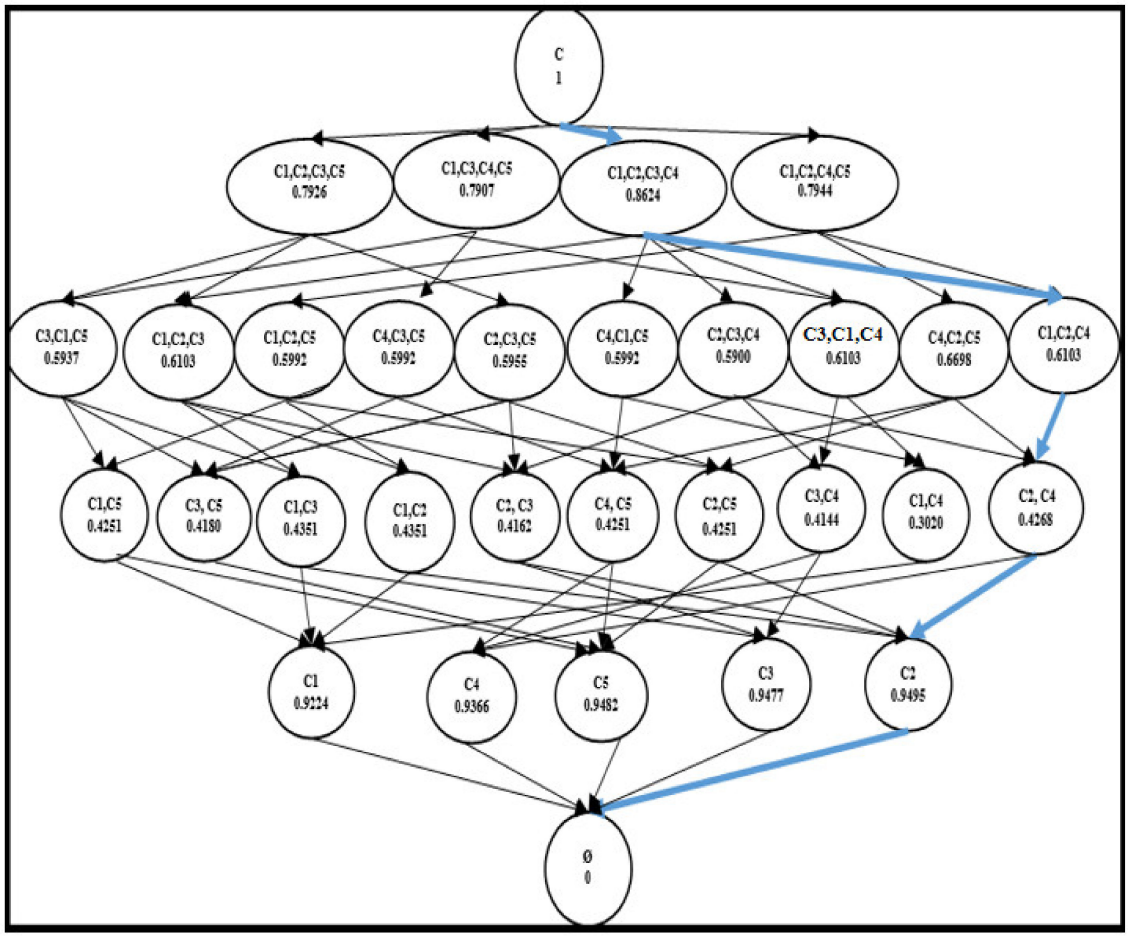

Figure 5. The fuzzy measure values of single classifiers and all possible sets of classifiers. The fuzzy measures utilized in the classification of the fourth Android app are illustrated by the blue paths.

Table 5 shows that the proposed Choquet integral-based ensemble of classifiers achieved the highest accuracy of 95.08%, which outperform that of single classifiers and arithmetic mean approach. Moreover, the proposed approach achieved the highest F1-score of 0.9350. It can be seen that the capability of the proposed approach to synthesize various classifiers classification outputs and take the benefits of each single classifier made it more effective, enabling it to achieve better performance and outperform the single classifiers and arithmetic mean approach. In addition to the basic measures listed above, the statistical significance of the proposed approach and other methods was measured. Figure 6 presents the critical difference diagram, where the models with statistically similar values of performance are connected to one another.

Table 5. Performance comparison between the proposed approach and other approaches.

Classifier	Accuracy	Precision	Recall	F1-Score
Random Forest	0.9495	0.8950	0.9680	0.9301
LGBM	0.9482	0.8921	0.9674	0.9282
Decision Tree	0.9480	0.8950	0.9639	0.9284
AdaBoost	0.9366	0.9251	0.9076	0.9163
XGBoost	0.9224	0.9192	0.8793	0.8988
Arithmetic Average	0.9486	0.9669	0.8940	0.92910
The proposed	0.9508	0.9240	0.9463	0.9350

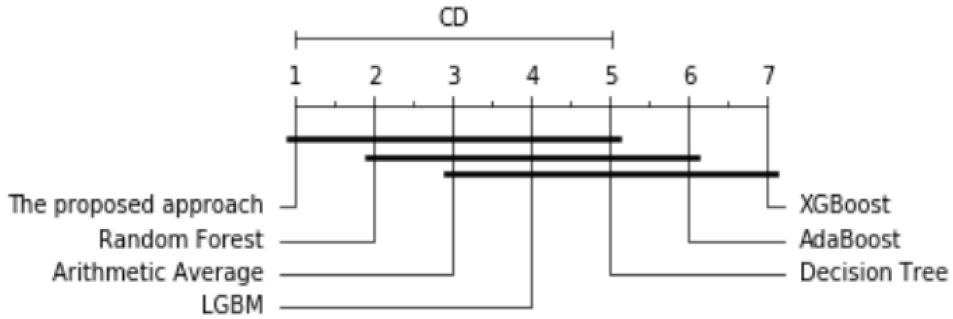

Figure 6. Critical difference diagram for the Nemenyi test.

The AUC is an important and useful evaluation of the overall performance [50]. An AUC with a high score indicates greater classification performance. As shown in Figure 7, the proposed approach achieved the highest AUC value of 95.0%, which demonstrates the ability of the proposed approach to differentiate between Android malware and goodware apps.

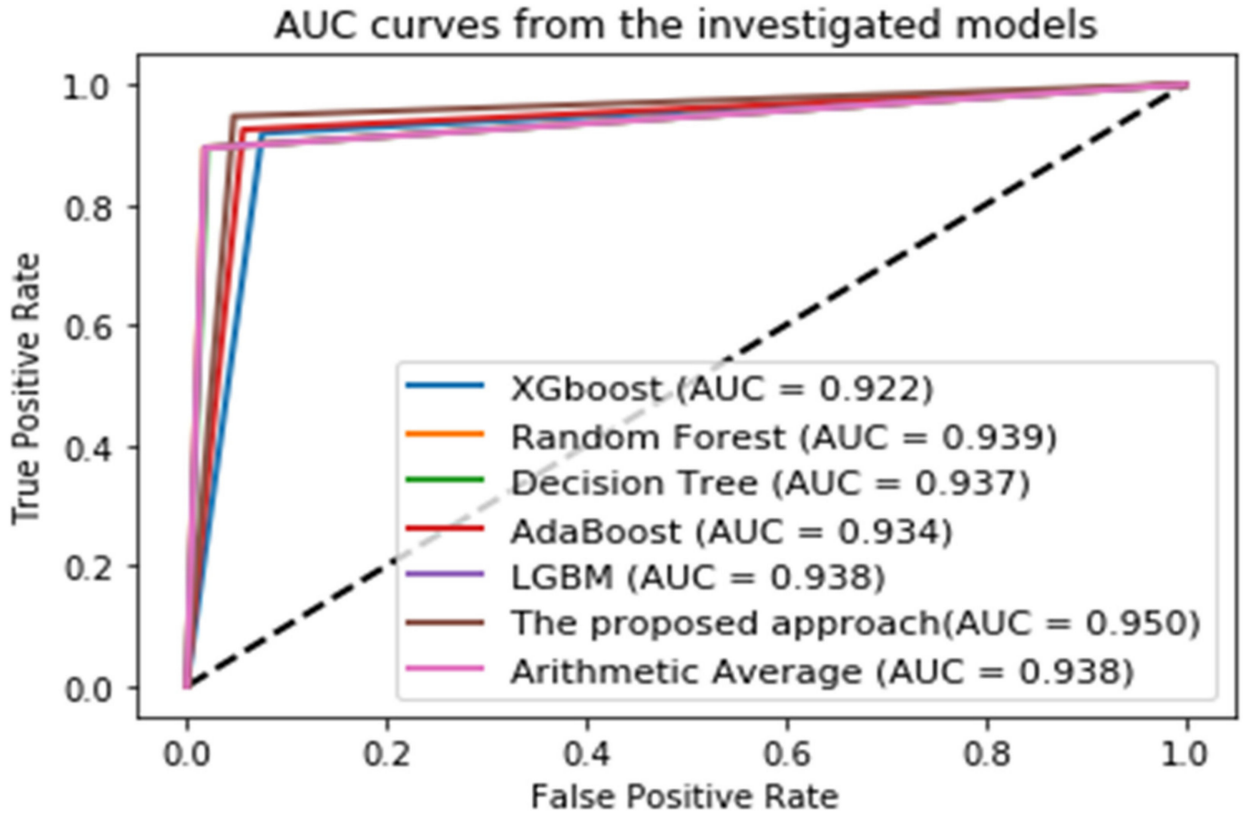

Figure 7. AUC curves for the proposed approach compared with other approaches.

In order to highlight the significance of our proposed approach, we compared its results with other published approaches in terms of accuracy. The suggested approach performs better than all the other approaches listed in Table 6, including our previous research [9,11,32], due to its ability to leverage the advantages of ensemble learning and adaptive fuzzy measures. Salah et al. [51] proposed the technique that attained the highest accuracy of 99%; however, they employed Android malware app features that differed from the features used in this study.

Table 6. Comparison between the proposed approach and other approaches.

Reference Approach	Accuracy
Altaher [9]	90%
Altaher and BaRukab [11]	91%
Abdulla and Altaher [10]	75%
Arp et al. [22]	94%
Firdaus et al. [23]	90%
Talha et al. [30]	88%
Taha and Malebary [32]	94.63%
Salah et al. [51]	99%
Yerima et al. [52]	93.1%
Sanz et al. [53]	85.8%
The proposed approach	95.08%

6. Conclusions

Android malware's continual growth poses a significant threat to the security of sensitive data and the privacy of smartphone users. It is critical to develop effective and fast methods for distinguishing Android malicious applications from legitimate applications. Android malware classification techniques based on multiple classifier fusion have been a growing trend in this field. This paper proposed a multi-classifier ensemble based on fuzzy integrals for detecting Android malware. The suggested technique combines and integrates the classification results of several classifiers, including XGBoost, Random Forest, Decision Tree, AdaBoost, and LightGBM, using the Choquet fuzzy integral as an aggregation function. The suggested approach can inherit the advantages of basic classifiers while avoiding their drawbacks. Additionally, the suggested technique is capable of taking into account classifier interactions. The experimental results indicate that combining several classifiers based on the Choquet fuzzy integers outperforms individual classifiers and achieves the highest accuracy of 95.08%. The limitation of the proposed approach is its inability to detect obfuscated Android malware apps. For future work, we aim to explore more advanced Android malware apps' features and techniques for integrating multiple classifiers to classify the Android malware apps.

Author Contributions: Conceptualization, A.T.; methodology, A.T.; implementation, A.T.; writing—original draft preparation, A.T.; writing—review and editing, O.B. and S.M. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number IFPHI-192-830-2020 and King Abdulaziz University, DSR, Jeddah, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Niu, W.; Cao, R.; Zhang, X.; Ding, K.; Zhang, K.; Li, T. OpCode-level function call graph based android malware classification using deep learning. *Sensors* **2020**, *20*, 3645. [CrossRef] [PubMed]
2. Statista. 2019. Available online: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store (accessed on 13 June 2021).

3. Gdata. 2019. Available online: https://www.gdata-software.com/news/g-data-mobile-malware-report-2019-new-high-for-malicious-android-apps (accessed on 13 April 2020).
4. Feng, J.; Shen, L.; Chen, Z.; Wang, Y.; Li, H. A two-layer deep learning method for android malware detection using network traffic. *IEEE Access* **2020**, *8*, 125786–125796. [CrossRef]
5. Conti, M.; Li, Q.Q.; Maragno, A.; Spolaor, R. The dark side (-channel) of mobile devices: A survey on network traffic analysis. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2658–2713. [CrossRef]
6. Mehtab, A.; Shahid, W.B.; Yaqoob, T.; Amjad, M.F.; Abbas, H.; Afzal, H.; Saqib, M.N. AdDroid: Rule-based machine learning framework for android malware analysis. *Mob. Netw. Appl.* **2020**, *25*, 180–192. [CrossRef]
7. Demontis, A.; Melis, M.; Biggio, B.; Maiorca, D.; Arp, D.; Rieck, K.; Corona, I.; Giacinto, G.; Roli, F. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE Trans. Depend. Secure Comput.* **2017**, *16*, 711–724. [CrossRef]
8. Papadopoulos, H.; Georgiou, N.; Eliades, C.; Konstantinidis, A. Android malware detection with unbiased confidence guarantees. *Neurocomputing* **2018**, *280*, 3–12. [CrossRef]
9. Altaher, A. An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features. *Neural Comput. Appl.* **2016**, *28*, 4147–4157. [CrossRef]
10. Abdulla, S.; Altaher, A. Intelligent approach for android malware detection. *KSII Trans. Internet Inf. Syst.* **2015**, *9*, 2964–2983.
11. Altaher, A.; Barukab, O. Android malware classification based on ANFIS with fuzzy c-means clustering using significant application permissions. *Turk. J. Electr. Eng. Comput. Sci.* **2017**, *25*, 2232–2242. [CrossRef]
12. Imtiaz, S.I.; Rehman, S.U.; Javed, A.R.; Jalil, Z.; Liu, X.; Alnumay, W.S. DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network. *Futur. Gener. Comput. Syst.* **2020**, *115*, 844–856. [CrossRef]
13. Wu, W.C.; Hung, S.H. DroidDolphin: A dynamic Android Malware Detection Framework using Big Data and Machine Learning. In Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems, Towson, MD, USA, 5–8 October 2014; pp. 247–252.
14. Burguera, I.; Zurutuza, U.; Nadjm-Tehrani, S. Crowdroid: Behavior-Based Malware Detection System for Android. In Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, Chicago, IL, USA, 15–19 November 2011; pp. 15–26.
15. Yang, Y.; Wei, Z.; Xu, Y.; He, H.; Wang, W. Droidward: An effective dynamic analysis method for vetting android applications. *Cluster Comput.* **2018**, *21*, 265–275. [CrossRef]
16. Dong, X.; Yu, Z.; Cao, W.; Shi, Y.; Ma, Q. A survey on ensemble learning. *Front. Comput. Sci.* **2019**, *14*, 241–258. [CrossRef]
17. Breiman, L. Bagging predictors. *Mach. Learn* **1996**, *24*, 123–140. [CrossRef]
18. Schapire, R.E.; Singer, Y. Improved Boosting Algorithms Using Confidence-rated Predictions. *Mach. Learn.* **1999**, *37*, 297–336. [CrossRef]
19. Wolpert, D.H. Stacked generalization. *Neural Netw.* **1992**, *5*, 241–259. [CrossRef]
20. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
21. Wang, W.; Li, Y.; Wang, X.; Liu, J.; Zhang, X. Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers. *Futur. Gener. Comput. Syst.* **2018**, *78*, 987–994. [CrossRef]
22. Arp, D.; Spreitzenbarth, M.; Hubner, M.; Gascon, H.; Rieck, K.; Siemens, C.E.R.T. Drebin: Effective and explainable detection of android malware in your pocket. *Ndss* **2014**, *14*, 23–26.
23. Firdaus, A.; Anuar, N.B.; Ab Razak, M.F.; Sangaiah, A.K. Bio-inspired computational paradigm for feature investigation and malware detection: Interactive analytics. *Multimed. Tools Appl.* **2017**, *77*, 17519–17555. [CrossRef]
24. Allix, K.; Bissyandé, T.F.; Klein, J.; Le Traon, Y. Androzoo: Collecting Millions of Android Apps for the Research Community. In Proceedings of the 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR) 2016, Austin, TX, USA, 14–15 May 2016; pp. 468–471.
25. Hu, D.; Ma, Z.; Zhang, X.; Li, P.; Ye, D.; Ling, B. The concept drift problem in Android malware detection and its solution. *Secur. Commun. Netw.* **2017**, 1–13. [CrossRef]
26. Zhang, X.; Hu, D.; Fan, Y.; Yu, K. A Novel Android Malware Detection Method Based on Markov Blanket. In Proceedings of the 2016 IEEE First International Conference on Data Science in Cyberspace (DSC) 2016, Changsha, China, 13–16 June 2016; pp. 347–352.
27. Coronado-De-Alba, L.D.; Rodríguez-Mota, A.; Escamilla-Ambrosio, P.J. Feature Selection and Ensemble of Classifiers for Android Malware Detection. In Proceedings of the 8th IEEE Latin-American Conference on Communications (LATINCOM), Medellin, Colombia, 15–17 November 2016; pp. 1–6.
28. Peiravian, N.; Zhu, X. Machine Learning for Android Malware Detection using Permission and Api Calls. In Proceedings of the 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Washington, DC, USA, 4–6 November 2013; pp. 300–305.
29. Wang, X.; Wang, W.; He, Y.; Liu, J.; Han, Z.; Zhang, X. Characterizing Android apps' behavior for effective detection of malapps at large scale. *Futur. Gener. Comput. Syst.* **2017**, *75*, 30–45. [CrossRef]
30. Talha, K.A.; Alper, D.I.; Aydin, C. APK Auditor: Permission-based Android malware detection system. *Digit. Investig.* **2015**, *13*, 1–14. [CrossRef]
31. Milosevic, N.; Dehghantanha, A.; Choo, K.-K.R. Machine learning aided Android malware classification. *Comput. Electr. Eng.* **2017**, *61*, 266–274. [CrossRef]

32. Taha, A.A.; Malebary, S.J. Hybrid classification of Android malware based on fuzzy clustering and the gradient boosting machine. *Neural Comput. Appl.* **2020**, *33*, 6721–6732. [CrossRef]
33. Awan, M.J.; Masood, O.A.; Mohammed, M.A.; Yasin, A.; Zain, A.M.; Damaševičius, R.; Abdulkareem, K.H. Image-Based Malware Classification Using VGG19 Network and Spatial Convolutional Attention. *Electronics* **2021**, *10*, 2444. [CrossRef]
34. Hemalatha, J.; Roseline, S.A.; Geetha, S.; Kadry, S.; Damaševičius, R. An efficient DenseNet-based deep learning model for malware detection. *Entropy* **2021**, *23*, 344. [CrossRef]
35. Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševičius, R.; Blažauskas, T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. *Appl. Sci.* **2020**, *10*, 4966. [CrossRef]
36. Choquet, G. Theory of Capacities. *Ann. Inst. Fourier* **1954**, *5*, 131–295. [CrossRef]
37. Höhle, U. Integration with Respect to Fuzzy Measures. In Proceedings of the IFAC Symposium on Theory and Applications of Digital Control, New Delhi, India, 5–7 January 1982; pp. 35–37.
38. Murofushi, T.; Sugeno, M. A theory of fuzzy measures: Representations, the Choquet integral, and null sets. *J. Math. Anal. Appl.* **1991**, *159*, 532–549. [CrossRef]
39. Murofushi, T.; Sugeno, M. An interpretation of fuzzy measures and the Choquet integral as an integral with respect to a fuzzy measure. *Fuzzy Sets Syst.* **1989**, *29*, 201–227. [CrossRef]
40. Li, X.; Wang, F.; Chen, X. Support Vector Machine Ensemble Based on Choquet Integral for Financial Distress Prediction. *Int. J. Pattern Recognit. Artif. Intell.* **2015**, *29*. [CrossRef]
41. Chiou, H.-K.; Tzeng, G.-H. Fuzzy Multiple-Criteria Decision-Making Approach for Industrial Green Engineering. *Environ. Manag.* **2002**, *30*, 816–830. [CrossRef]
42. Tahani, H.; Keller, J.M. Information fusion in computer vision using the fuzzy integral. *IEEE Trans. Syst. Man Cyber.* **1990**, *733*, 741. [CrossRef]
43. Mori, T. Information Gain Ratio as Term Weight: The Case of Summarization of Ir Results. In Proceedings of the COLING 2002, The 19th International Conference on Computational Linguistics, Taipei, Taiwan, 26–30 August 2002.
44. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM S International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016; pp. 785–794.
45. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
46. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*; Routledge: London, UK, 2017.
47. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [CrossRef]
48. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 3146–3154.
49. Chibelushi, C.; Deravi, F.; Mason, J. Adaptive classifier integration for robust pattern recognition. *IEEE Trans. Syst. Man Cybern. Part B* **1999**, *29*, 902–907. [CrossRef] [PubMed]
50. Dal Pozzolo, A.; Caelen, O.; Le Borgne, Y.-A.; Waterschoot, S.; Bontempi, G. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Syst. Appl.* **2014**, *41*, 4915–4928. [CrossRef]
51. Salah, A.; Shalabi, E.; Khedr, W. A lightweight android malware classifier using novel feature selection methods. *Symmetry* **2020**, *12*, 858. [CrossRef]
52. Yerima, S.Y.; Sezer, S.; McWilliams, G. Analysis of Bayesian classification-based approaches for Android malware detection. *IET Inf. Secur.* **2013**, *8*, 25–36. [CrossRef]
53. Sanz, B.; Santos, I.; Laorden, C.; Ugarte-Pedrero, X.; Bringas, P.G.; Álvarez, G. Puma: Permission Usage to Detect Malware in Android. In *International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 289–298.