ASSESSMENT OF THE DETECTION CAPACITY OF NORMALIZED GAMMACHIRP CEPSTRAL COEFFICIENTS IN ANDROID MALWARE

Paul Tarwireyi
Research Scholar, Department of Computer Science, University of Zululand
KwaDlangezwa, 3880, South Africa
tarwireyip@unizulu.ac.za
https://orcid.org/0000-0001-6625-3826

Alfredo Terzoli
Professor, Department of Computer Science, University of Zululand
KwaDlangezwa, 3880, South Africa
terzolia@unizulu.ac.za

Matthew O Adigun
Senior Professor, Department of Computer Science, University of Zululand
KwaDlangezwa, 3880, South Africa
adigunm@unizulu.ac.za

Abstract

Android has grown to become the leading mobile operating system on the market due, in part, to its popularity and open-source nature. On the other hand, the Android ecosystem has become a fertile ground for malware, posing substantial security risks to ordinary mobile users. This is hardly surprising given the sheer number of devices it controls. Despite several research efforts over the years, Android malware detection remains a challenge. This is largely due to the fact that it is still unknown which features or combination of features can effectively distinguish malicious Android applications from benign ones. To this end, this research explores an Android malware detection system that uses the low-level audio Normalized Gammachirp Cepstral Coefficients as features to classify malware with machine learning techniques. First, we convert the Android Application Package datasets into audio datasets, then extract the audio features. To evaluate our approach, twenty-four machine learning algorithms were implemented, and results were collected. The experimental results show that the proposed malware features achieved the highest accuracy, precision, recall, f1-score and AUC and with respective values of 97.6%, 98.3%, 98.6%, 98.4% and 96.4%%. It also achieved 98% area under the precision-recall curve indicating that Normalized Gammachirp Cepstral Coefficients are effective for Android malware classification. Moreover, the processing and detection times were also reasonably short. Amongst the best-performing models were Extratrees, Random forest, CatBoost, XGBoost and KNeighbors.

Keywords: **Android malware; Classification; Normalized Gammachirp Cepstral Coefficients; malware audio.**

1. Introduction

Mobile devices have become an attractive target for malware attackers owing to their wide usage and them being a wealthy source of valuable personal information. Furthermore, the pandemic has forced many to adopt remote working, resultantly, mobile devices also started storing an increasing amount of business information. Furthermore, the associated rise in employee mobility widened the mobile device attack surface. For the past few years, malware attackers and vendors have been engaged in a back-and-forth battle in the Android ecosystem, resembling a game of cat and mouse. In no time the vendors have created signatures to detect and mitigate known attacks, attackers are simply tweaking some code to alter or hide the signature and then relaunch the attack. This has left malware defenders with few options other than to default to the defensive mode where they are pretty much reactionary to the emerging threats. In this mode, security experts have to handcraft malware signatures which are then stored in databases and used by anti-malware solutions for detection [1]. Due to the daily influx of new malware being generated, it has been challenging for malware security experts to generate signatures quickly enough to counteract the security threat. Thus, it has become common knowledge that traditional malware detection mechanisms are ineffective in detecting zero-day attack [2]. In response, researchers have begun investigating the use of machine learning techniques to automate the identification and detection of malware.

Due to its success in other disciplines, machine learning has a huge potential to curb the affliction of malware. However, its application in the malware detection domain is not straightforward as it necessitates the use of appropriate application characteristics with high discriminative capabilities to achieve higher prediction rates. The question of finding such characteristics has not yet been settled in this research area.

The purpose of this work is to present the application of a biologically inspired feature that was proposed for speaker identification and other audio classification problems - Normalized Gammachirp Cepstral Coefficients, for android malware detection. To the best of our knowledge, no prior research has investigated the viability of Normalized Gammachirp Cepstral Coefficients as a malware detection feature in Windows, Linux, or Android.

Our contributions can be summarized as follows:

- As there are no existing publicly available audio-based Android malware datasets, we start by creating such datasets.
- Secondly, as there are no existing publicly available Normalized Gammachirp Cepstral Coefficients classification datasets for android malware detection, we also create such datasets that can be used directly with existing machine learning algorithms for malware detection.
- We propose an Android malware detection system based on Normalized Gammachirp Cepstral Coefficients.
- The malware detection problem is transformed into an audio signal classification problem.

2. Literature review

The typical techniques found in the literature for addressing this problem are static-based techniques that utilize requested permissions [3]–[6], API calls [7]–[11], system calls [12], [13], operation code sequences [14], app rating, number of times downloaded, and hashing as features that can be used for android malware classification. Most of the efforts in the research community have gone into solving this problem by using Natural language processing techniques such as N-grams.

Lekssays [15] presents a static analysis approach that converts the android application packages into gray-scale images, then uses deep learning and convolutional neural networks. They tried different architectures with this visual analysis approach using the CICAndMal2017 dataset and achieved an accuracy of 84.9%.

[16] proposed work that mapped the PE header part of Windows binary files into MIDI messages. The MIDI messages were then synthesized into audio signals. They extracted chromagram and MFCCs for classification with KNN, SVM, Adaboost, and Random Forest. They assessed their system on the Kaggle Microsoft Malware Classification Challenge (BIG 2015) dataset. They achieved an accuracy score of 96.1%.

Authors in [17] extracted dex files from apks and converted them into wav files. From the audio files they then extracted chromagram, root mean square, spectral centroid, spectral rolloff, zero crossing rate, poly, tonnetz and MFCC features to create a feature vector. The features were evaluated using WEKA for kNN, SVM, LogisticRegression. The Neural network was implemented using python. Using a dataset of 50000 apps, they achieved a binary classification accuracy score of 95.2 and family classification accuracy of 92.2%. Malicious files were obtained from the AMD dataset while benign files were crawled from google play.

In a follow up paper from similar authors in [18], a method for malicious family detection exploiting audio signal processing was also presented. They extracted the same features they had used in the previous paper. However, using an Android dataset with 4746 samples belonging to 10 families. Samples were collected from AMD (no longer available), contagio and Drebin. They obtained 98.8% accuracy using a deep learning model.

Although some progress has been made using these features, further research is needed. Few researchers have investigated the use of image processing for malware detection, and only very recently have they begun to investigate the application of automated signal processing techniques to Android detection or malware detection in general. Currently, there is little published literature on audio techniques in malware classification. This provides an opportunity to explore this new and relatively unexplored area of malware feature engineering and detection.

This work explores the feasibility of a biologically inspired noise-robust algorithm that has fared well in the audio signal processing domain, android malware detection. For a very long time, Mel frequency spectrum coefficient features have been used as the defacto speaker cognition standard. In the MFCC method, a short-time spectral analysis is used to extract spectrum information from short excerpts of the signal [19], [20].

Prior studies have shown that MFCCs perform very well in speaker identification under clean sound conditions however their performance degrades by more than 50% if the input signal is very noisy [21]. This led to researchers looking for more algorithms that can perform very well under noisy conditions. One such algorithm, Gammachirp auditory filter bank, was proposed by Irino and Patterson [22], [23] to improve the simulation of the basic perceptual and psychophysics of the basilar membrane [21], [24]. These filter banks were successfully used to extract noise robust auditory in many audio signal processing areas such as speaker identification and speech recognition [25] [26] [27]; [28] .

2.1. Peripheral Auditory System

For humans to be able to comprehend speech, the auditory system uses complex, multi-stage processing of the auditory information received from the environment [22]. First, the outer part of the peripheral auditory system picks sound signals from the environment, which are then filtered and channeled through the ear canal to the middle ear. The middle ear transforms the signal and further transmits it to the inner ear, where the signal is processed further and converted to electrical signals for onward transmission to the brain [29]. The peripheral auditory model is the mathematical model used to simulate the complex auditory information processing carried out by the peripheral auditory system [21], [29].

2.2. Gammachirp auditory filter

The Gammachirp auditory filter bank was proposed by Irino and Patterson [22]–[24] to improve the simulation of the basic perceptual and psychophysics of the basilar membrane [24]. These filter banks were successfully used to extract noise robust auditory in many audio signal processing areas such as speaker identification and speech recognition [25]–[28].

The gammachirp filter bank has an impulse response of the following classical form:

$$g_c(t) = at^{n-1}\exp(-2\pi bEBR(f_r)t)\exp(j2\pi f_r t + jc\ln t + jc\phi) \tag{1}$$

Where time $t > 0$, C is the chirp rate, a is the amplitude, n and b are parameters defining the envelope of the gamma distribution, ϕ is the initial phase and f_r is the asymptotic frequency. ln t is a natural logarithm of time, and $ERB(f_r)$ represents the equivalent rectangular bandwidth of the Gammachirp auditory filter at moderate levels in Hz [23], [24]. The Fourier transform of the gammachirp represented in equation (4) is derived as follows [24]:

$$|G_c(f)| = \frac{a|\Gamma(n + jc)|e^{c\theta}}{(2\pi)^n\left[\left(bERB(f_r)\right)^2 + (f - f_r)^2\right]^{\frac{n}{2}}} \tag{2}$$

Where $\theta = \arctan\left(\frac{f-f_r}{bERB(f_r)}\right)$ and $\Gamma(n + jc)$ is the complex gamma distribution.

3. Method

This section presents the methodological steps that were followed to realize the proposed Normalized Gammachirp Cepstral Coefficients malware detection system. The following steps were taken in the work.

(1) The experimental android application package datasets were collected the stored for preprocessing

(2) The first preprocessing step was to convert the apk files into audio following a procedure followed in our initial paper [30].

(3) Like the APK files, which could not be directly injected into machine learning models, the Normalized Gammachirp Cepstral Coefficients features were extracted from the audio files as the next preprocessing step. Further details regarding the feature extraction process will follow shortly. Our work took this route due to a number of considerations. Firstly, even though a few attempted using raw audio files for audio classification [31], [32], they relied on very deep convolutional neural networks that operated on same-length audio files, required more resources and the results are not yet outperforming the traditional low-level hand-crafted features. In our case, the lengths of the malware audio samples vary.

(4) The resultant dataset of extracted features is split into training and testing sets.

(5) Twenty-four machine learning algorithms are created, trained, and validated to learn the intrinsic patterns that can be used to distinguish between malicious and benign android application package files.

(6) The trained classifiers are evaluated using the test set and used to make predictions on data it has never seen before.

(7) Evaluation results comparing the twenty-four machine learning algorithms based on eight metrics namely, sensitivity, specificity, accuracy, recall, F1-score, AUC, training, and testing time are collected and analyzed.

(8) The best performing model is identified

3.1. Extracting Normalized Gammachirp Cepstral Coefficients

Normalized Gammachirp Cepstral Coefficients - the technique lying at the heart of our approach, was originally proposed as a bio-inspired feature extraction method for noise-robust speech recognition in the audio engineering field in [22], [23] and was applied in many works such [33] with relatively good results.

To extract features, pre-emphasis is applied to the audio signal as a filter to compensate for the average spectral shape. A twenty-five-millisecond hamming windowing is used to split the input signal into short enough temporal segments, which does not allow enough time for the properties of the signal to change in each segment [21]. The signal is passed through the Fast Fourier transform to obtain a short-term power spectrum [34]. The resultant signal is passed through a second-order filter and a normalized gammachirp auditory filter bank. The filter bank outputs are logarithmically compressed in order to mode the loudness perception in the human auditory system. The signal is passed through the discrete cosine transform, a time-frequency transform operation for decorrelating sequentially correlated data and outputting the Normalized Gammachirp Cepstral Coefficients [21]. Figure 1 shows the block diagram for normalized gammachirp Cepstral Coefficients algorithm.

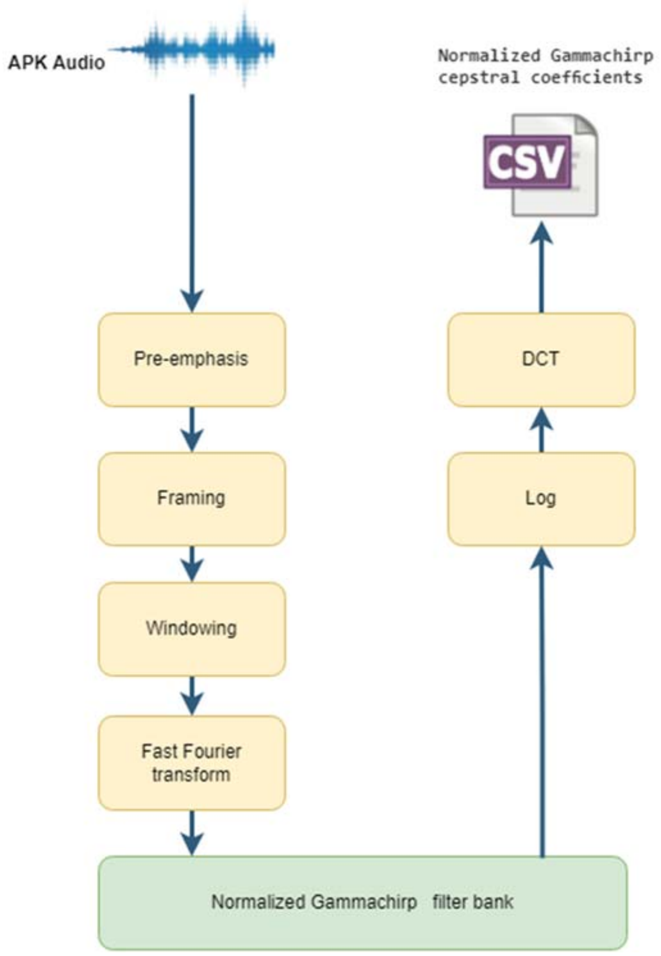
APK Audio
Normalized Gammachirp cepstral coefficients
CSV
Pre-emphasis
Framing
Windowing
Fast Fourier transform
Normalized Gammachirp filter bank
Log
DCT

Fig.1. Normalized Gammachirp Cepstral Coefficients algorithm block diagram [21]

3.2. *Datasets*

We collected malicious and benign application samples from the Canadian Institute for Cybersecurity (CIC), CICAndMal2017 [35], and CICMalDroid 2020 [36], [37]. While the CICAndMal2017 has 2126 apks, the CICMalDroid 2020 dataset is composed of 17341 android application packages collected from several sources such as Contagio blog, AMD, Maldozer, and other recent and sophisticated datasets collected until 2018.

3.3. *Evaluation metrics*

The chosen machine learning classification models are evaluated by calculating several evaluation metrics. We train the twenty-four machine learning models on the training set and evaluate them on the testing set. We evaluate the performance of these models using accuracy, precision, recall, f1 score, and Area Under the Curve determined by Eq. (3) to (6). Furthermore, precision-recall curves, train time, and testing time are also used.

Accuracy is determined by the ratio of the total number of instances that are correctly predicted, over the total number of all instances. It is calculated using eq. 3 below

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3}$$

Precision represents the ratio of true positives to all positive predictions. It is calculated using eq. 4 below

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

The recall represents a ratio of the total number of positive predictions to the total number of all positive samples. This is also known as sensitivity or true positive rate. It is calculated using eq. 5 below

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

F1-score is the harmonic mean of the precision and recall. It is calculated using eq. 6 below

$$F1 - score = 2\,x\,\frac{Precision\,x\,Recall}{Precision\,x\,Recall} \tag{6}$$

The definitions of FP, TN, TP, and FN in the formulas are given as follows:

- False positive (FP) is the Type I error or a false alarm. It represents the number of benign applications which are incorrectly classified as malicious.
- True negative (TN) stands for the number of benign applications which are correctly classified as benign.
- True positive (TP) is the number of malicious applications that are correctly classified as malicious.
- False negative (FN) is the Type II error which represents the number of malicious applications that are inaccurately classified as benign.

4. Results and Discussion

In this section, evaluation experiments for the detection and classification of malware on the CICAndMal2017 and CICMalDroid2020 datasets are carried out, and the findings are presented.

The experimental setup was done on a 1.80GHz Intel(R) Core (TM) i7-8565U CPU laptop with 24 GB RAM, 64-bit Windows operating system, and NVIDIA GeForce MX 130 GPU (Graphics Processing Unit). The code to implement the machine learning pipeline discussed in the methodology was developed using TensorFlow and Python.

Converting CICMaldroid2020 apks to waveform audio files took approximately 21 minutes, whereas CICAndMal2017 took approximately 5 minutes and 34 seconds. Twenty-four classic machine learning algorithms were implemented for performance evaluation. These include extra trees, lightGBM, Gaussian Process, Multi-Layer Perceptron, CatBoost, Bayesian Network, Passive Aggressive, Support Vector Machine, AdaBoost, Random Forest, KNeighbors, and Decision Tree.

4.1. *CIAndMal2017 Classification performance*

Table 1 below displays the results of the malware detection and classification experiments on the CICMalAnal2017 dataset using an 80 – 20 % split.

Classifier used	Test Accuracy	Precision	Recall	F1 Score	AUC	Train Time [s]	Test Time [s]
RandomForest	0.9349	0.8919	0.7765	0.8302	0.8761	2.6614	0.0676
ExtraTrees	0.9301	0.8889	0.7529	0.8153	0.8643	0.2942	0.0177
LGBM	0.9229	0.8442	0.7647	0.8025	0.8642	0.167	0.002
XGB	0.9205	0.8333	0.7647	0.7975	0.8627	0.5566	0.0034
CatBoost	0.9181	0.84	0.7412	0.7875	0.8524	3.8794	0.0033
GradientBoosting	0.9181	0.8696	0.7059	0.7792	0.8393	0.6371	0.0011
MLP	0.9157	0.8571	0.7059	0.7742	0.8378	2.0255	0.001
GaussianProcess	0.9157	0.8472	0.7176	0.7771	0.8422	2.7054	0.035
XGBRF	0.9108	0.875	0.6588	0.7517	0.8173	0.2468	0.0041
Bagging	0.9108	0.8243	0.7176	0.7673	0.8391	0.1517	0.0023
KNeighbors	0.906	0.8194	0.6941	0.7516	0.8274	0.0026	0.0168
SVC	0.9036	0.8571	0.6353	0.7297	0.804	0.2474	0.0117
SGD	0.8916	0.8333	0.5882	0.6897	0.779	0.006	0.0002
RidgeCV	0.8892	0.8679	0.5412	0.6667	0.76	0.0122	0.0003
LogisticRegressionCV	0.8867	0.8167	0.5765	0.6759	0.7716	0.4218	0.0003
LinearSVC	0.8867	0.8393	0.5529	0.6667	0.7628	0.0766	0.0003
AdaBoost	0.8867	0.7969	0.6	0.6846	0.7803	0.2146	0.0095
LogisticRegression	0.8867	0.8167	0.5765	0.6759	0.7716	0.0157	0.0003
NuSVC	0.8819	0.7	0.7412	0.72	0.8297	0.2727	0.0082
DecisionTree	0.8747	0.6988	0.6824	0.6905	0.8033	0.0235	0.0003
PassiveAggressive	0.8554	0.6316	0.7059	0.6667	0.7999	0.0038	0.0003
BernoulliNB	0.8265	0.6102	0.4235	0.5	0.6769	0.0023	0.0005
GaussianNB	0.8024	0.5231	0.4	0.4533	0.653	0.0021	0.0005
Perceptron	0.689	0.7108	0.3423	0.4471	0.3878	0.6129	0.0026

Table 1. CICAndMal2017 performance results.

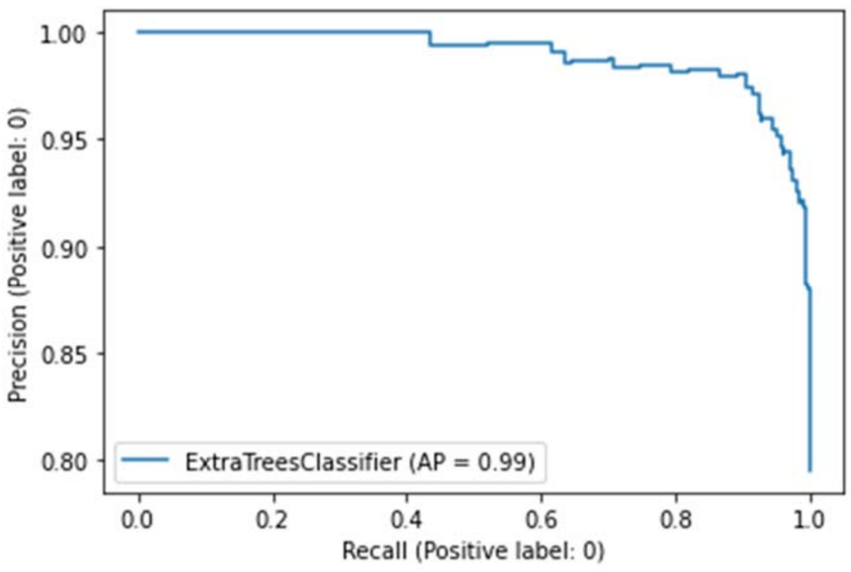

Fig. 2. Extra Trees Precision-recall curve with CICAndMal2017 dataset

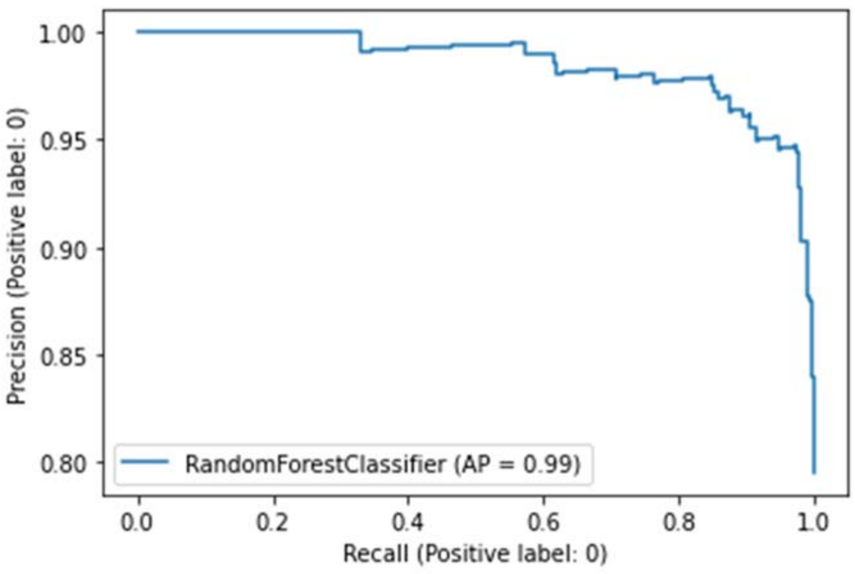

Fig. 3. Random Forest Precision-recall curve with CICAndMal2017 dataset

It can be observed in fig. 2 and fig. 3 that the proposed malware detection method achieved 99% area under the precision-recall curve with the Extra Trees and Random Forest. This curve is used for evaluating the performance of binary classification algorithms with imbalanced datasets.

Out of the machine learning algorithms mentioned in Table 3, the following can be observed for the top 3 results.

- Accuracy: Random Forest, extra trees, and lighGBM algorithms achieved the highest scores of 93.5%, 93.0%, and 92.3%, respectively. Fig. 4 shows the complete overview of how the algorithms performed in terms of accuracy.

- Precision: Random Forest, extra trees, and GradientBoosting algorithms attained highest scores of 89.2%, 88.9%, and 87%, respectively.

- Recall: Random Forest, lighGBM, and XGBoost algorithms reached highest scores of 77.7%, 76.5%, and 76.5%, respectively.

- F1 Score: Random Forest, extra trees, and lighGBM algorithms achieved the highest scores of 83%, 81.5%, and 80.3%, respectively.

- ROC Score: Random Forest, extra trees, and lighGBM algorithms achieved the highest scores of 87.6%, 86.4%, and 86.4%, respectively.

In terms of training and testing time, it was observed that lightGBM performed better taking only 0.167 seconds for training and 0.002 seconds for testing. This makes it a good candidate for on-device malware detection model.

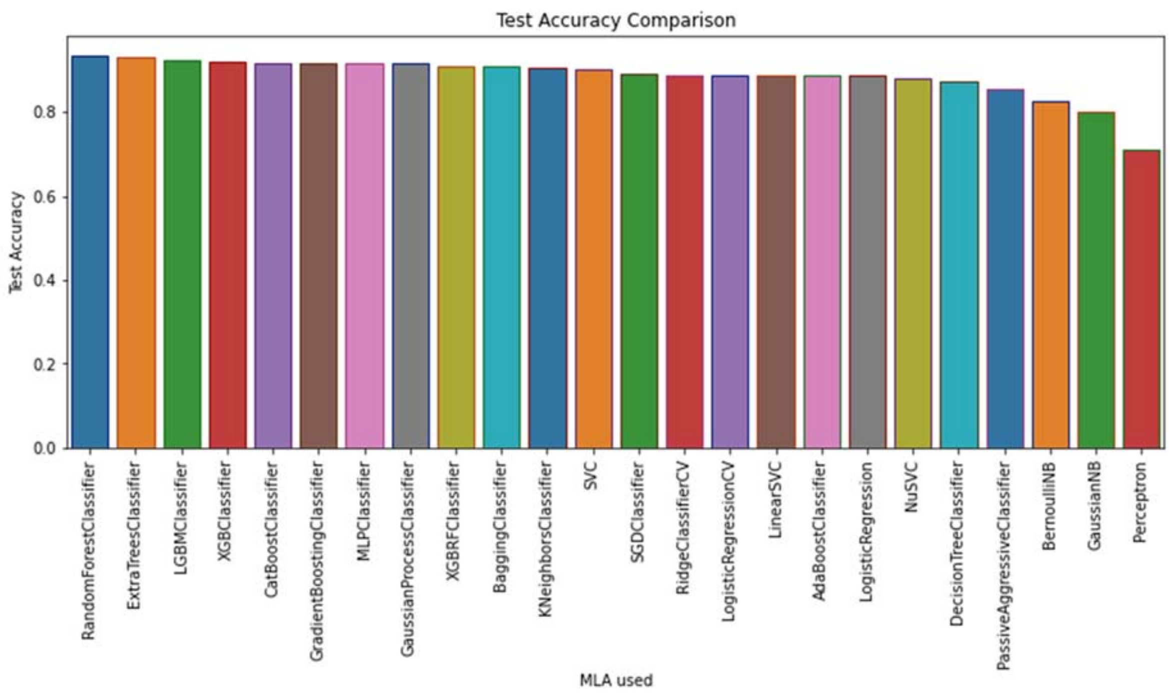

Fig. 4. Test accuracies achieved with the CICAndMal2017 dataset

In Fig. 4. we compare the accuracy performance metric for the twenty-four classifiers. It can be noted that ensembles generally perform well in this task, whole the perceptron had the worst results.

4.2. *CIMalDroid2020 Classification performance*

Table 4 shows the experimental results of the various models that were evaluated in this study under the CICMaldroid2020 dataset using an 80 – 20 % split.

MLA used	Test Accuracy	Precision	Recall	F1 Score	AUC	Train Time [s]	Test Time [s]
ExtraTrees	0.9759	0.9826	0.9861	0.9844	0.964	1.7178	0.0547
RandomForest	0.9743	0.9806	0.9861	0.9834	0.9605	20.2133	0.2004
CatBoost	0.9735	0.9826	0.9831	0.9828	0.9624	6.0953	0.0118
XGB	0.9704	0.98	0.9815	0.9808	0.9574	2.1929	0.005
LGBM	0.968	0.9775	0.981	0.9793	0.9528	0.2019	0.006
KNeighbors	0.9657	0.9784	0.9769	0.9777	0.9525	0.0393	0.1564
MLP	0.9633	0.9788	0.9733	0.9761	0.9516	17.1286	0.0017
Bagging	0.9617	0.9724	0.9779	0.9752	0.9427	1.3413	0.0061
GaussianProcess	0.9562	0.9703	0.9728	0.9716	0.9367	530.9421	0.6772
DecisionTree	0.9518	0.9574	0.981	0.9691	0.9177	0.2544	0.0006
GradientBoosting	0.951	0.9616	0.9754	0.9684	0.9226	5.6972	0.0048
SVC	0.9471	0.9604	0.9713	0.9658	0.9189	10.3413	0.2791
LogisticRegressionCV	0.9435	0.9584	0.9687	0.9635	0.9141	2.1283	0.0004
LogisticRegression	0.9432	0.9584	0.9682	0.9632	0.9139	0.4858	0.0005
XGBRF	0.9416	0.9523	0.9728	0.9624	0.9051	1.5731	0.0056
SGD	0.9408	0.9592	0.9641	0.9616	0.9135	0.0795	0.0005
LinearSVC	0.9404	0.9559	0.9672	0.9615	0.9091	0.5831	0.0007
AdaBoost	0.9384	0.9553	0.9651	0.9602	0.9072	1.4058	0.0213
PassiveAggressive	0.9329	0.9597	0.9528	0.9562	0.9096	0.0185	0.0006
Perceptron	0.912	0.9177	0.9728	0.9445	0.8409	0.0145	0.0005
RidgeCV	0.906	0.9084	0.9764	0.9411	0.8238	0.0324	0.0009
NuSVC	0.8685	0.9048	0.9266	0.9156	0.8006	7.8621	0.1348
BernoulliNB	0.7639	0.878	0.805	0.8399	0.7159	0.0056	0.0008
GaussianNB	0.6636	0.9127	0.6224	0.7401	0.7119	0.0081	0.0013

Table 2. CICMaldroid2020 performance results.

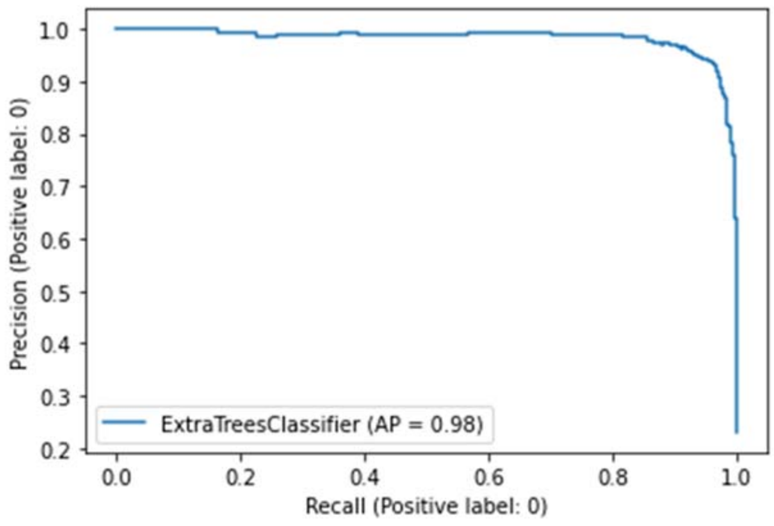

Fig. 5. Random Forest Precision-recall curve with CICMalDroid2020 dataset

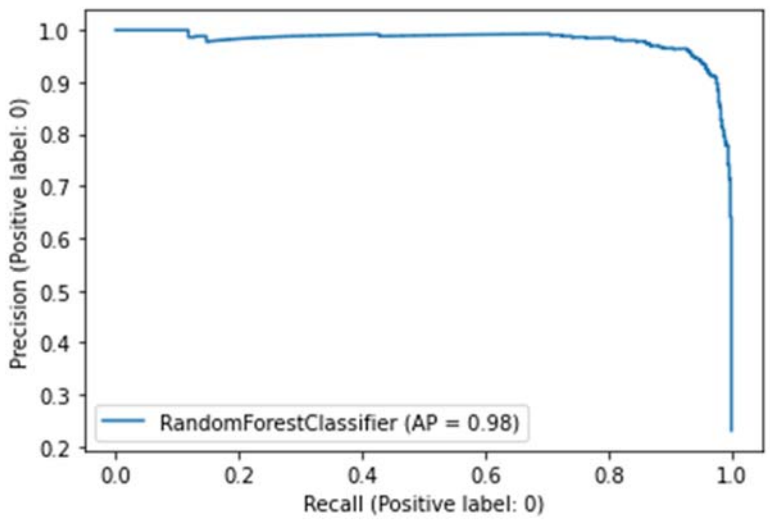

Fig. 6. Random Forest Precision-recall curve with CICMalDroid2020 dataset

The table is sorted by the testing accuracy column. As we can see from the results in table 4, for:

- Accuracy: Random Forest, extra trees and catboost algorithms achieved the top 3 test accuracy scores. The highest scores achieved are 97.6%, 97.4%, and 97.4%, respectively.
- Precision: Extra Trees achieved the highest score of 98.3%, followed by random forest which had 98.1%.
- Recall: Random forest and extra trees had a tie of 98.6% followed by catboost which had 98.3%.
- F1 Score: Random forest was the best with an f1 score of 98.4%, followed by extra trees which had 98.3%.
- ROC Score: Extra trees attained the best score of 96.4%, followed by random forest, which had 96.1%.

In Fig. 7. we graphically compare the accuracy performance metric for the twenty-four classifiers. It can be noted that ensembles also perform well in this task, whole the perceptron had the worst results.

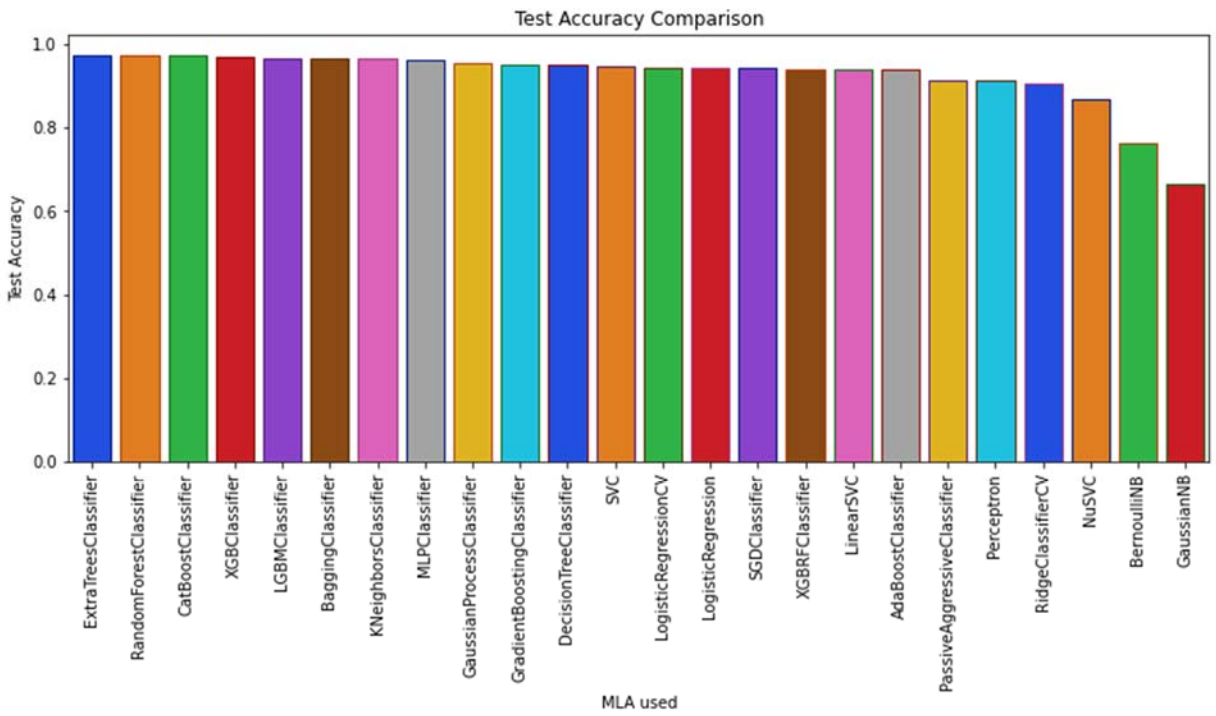

Figure 7. Test Accuracy Performance results for CICMalDroid2020

The table below ranks the models column-wise by identifying the best model according to each performance metric.

Classifier used	Test Accuracy	Precision	Recall	F1 Score	AUC	Train Time [s]	Test Time [s]
ExtraTrees	**1**	**1**	**1**	**1**	**1**	15	19
RandomForest	2	**3**	2	2	3	23	22
CatBoost	3	2	3	3	2	19	17
XGB	4	4	4	4	4	17	13
LGBM	5	7	5	5	5	8	15
KNeighbors	6	6	8	6	6	6	21
MLP	7	5	11	7	7	22	11
Bagging	8	8	7	8	8	12	16
GaussianProcess	9	9	12	9	9	24	24
DecisionTree	10	16	6	10	12	9	6
GradientBoosting	11	10	10	11	10	18	12
SVC	12	11	15	12	11	21	23
LogisticRegressionCV	13	15	16	13	13	16	1
LogisticRegression	14	14	17	15	14	10	4
XGBRF	15	19	13	14	19	14	14
SGD	16	13	20	16	15	7	3
LinearSVC	17	17	18	17	17	11	7
AdaBoost	18	18	19	18	18	13	18
PassiveAggressive	19	12	21	19	16	4	5
Perceptron	20	20	14	20	20	3	2
RidgeCV	21	22	9	21	21	5	9
NuSVC	22	23	22	22	22	20	20
BernoulliNB	23	24	23	23	23	1	8
GaussianNB	24	21	24	24	24	2	10

Table 3. CICMaldroid2020 performance ranking results.

Based on the highest test accuracy results, the top five models are Extra Trees, Random Forest, CatBoost, XGB and LGBM. Even though the extra trees algorithm was a clear top performer in all classic performance metrics including execution time, the precision-recall graphs shown in fig. 5. and fig. 6. show that the performance might be almost the same. Both algorithms have high precision and recall rates, meaning they returned a high number of correctly labelled results. From the analysis of Table 1 and Table 2, it can also be noted that the highest accuracy achieved improved by around 4.1% with the CICMalDroid2020 dataset compared to the CICAndMal2017 dataset. The results show that normalized gammachirp cepstral coefficients are promising static features for malware detection.

Conclusion

We propose an end-to-end, static android malware detection strategy that uses acoustic signals and Normalized Gammachirp Cepstral Coefficients as malware features. To the best of our knowledge, this is the first study to utilize such features for malware detection. The proposed approach first converts android application package files into audio. Normalized Gammachirp Cepstral Coefficients features are then extracted from the audio files and used as machine learning features. The findings show that the proposed audio feature may considerably detect malware with competitive accuracy levels, inspiring other researchers to develop stronger machine learning pipelines and models for intelligent malware detection in this direction. In future work, familial classification can be considered.

Data availability

The datasets generated and/or used for analysis during the current study are available from the corresponding author on reasonable request.

Conflicts of interest

The authors have no conflicts of interest to declare

References

[1] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, ‘A Review of Android Malware Detection Approaches Based on Machine Learning’, *IEEE Access*, vol. 8, pp. 124579–124607, 2020, doi: 10.1109/ACCESS.2020.3006143.

[2] A. Razgallah, R. Khoury, S. Hallé, and K. Khanmohammadi, ‘A survey of malware detection in Android apps: Recommendations and perspectives for future research’, *Comput Sci Rev*, vol. 39, p. 100358, Feb. 2021, doi: 10.1016/J.COSREV.2020.100358.

[3] H. H. R. Manzil and M. S. Naik, ‘COVID-Themed Android Malware Analysis and Detection Framework Based on Permissions’, *2022 International Conference for Advancement in Technology, ICONAT 2022*, 2022, doi: 10.1109/ICONAT53423.2022.9726024.

[4] O. E. Kural, D. Ö. Şahin, S. Akleylek, and E. Kılıç, ‘Permission Weighting Approaches in Permission Based Android Malware Detection’, in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, 2019. Accessed: Aug. 01, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/8907187/

[5] D. Ö. Şahın, O. E. Kural, S. Akleylek, and E. Kiliç, ‘New results on permission based static analysis for Android malware’, in *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, 2018. Accessed: Aug. 01, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/8355377/

[6] K. Santosh Jhansi, S. Chakravarty, and R. K. P. Varma, ‘Feature Selection and Evaluation of Permission-based Android Malware Detection’, *Proceedings of the 4th International Conference on Trends in Electronics and Informatics, ICOEI 2020*, pp. 795–799, Jun. 2020, doi: 10.1109/ICOEI48184.2020.9142929.

[7] S. Kumar, D. Mishra, and S. K. Shukla, ‘Android Malware Family Classification: What Works-API Calls, Permissions or API Packages?’, *Proceedings - 2021 14th International Conference on Security of Information and Networks, SIN 2021*, 2021, doi: 10.1109/SIN54109.2021.9699322.

[8] N. Nsuworks and F. Guyton, ‘Feature Selection on Permissions, Intents and APIs for Android Feature Selection on Permissions, Intents and APIs for Android Malware Detection Malware Detection’. [Online]. Available: https://nsuworks.nova.edu/gscis_etd

[9] K. Iwamoto and K. Wasaki, ‘Malware classification based on extracted API sequences using static analysis’, in *Asian Internet Engineeering Conference, AINTEC 2012*, 2012, pp. 31–38. doi: 10.1145/2402599.2402604.

[10] F. Guyton, W. Li, L. Wang, and A. Kumar, ‘Android Feature Selection based on Permissions, Intents, and API Calls’, *2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA)*, pp. 149–154, May 2022, doi: 10.1109/SERA54885.2022.9806471.

[11] J. Park, H. Chun, and S. Jung, ‘API and permission-based classification system for Android malware analysis’, in *2018 International Conference on Information Networking (ICOIN)*, 2018. Accessed: Aug. 01, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/8343260/

[12] A. Ananya, A. Aswathy, T. R. Amal, P. G. Swathy, P. Vinod, and S. Mohammad, ‘SysDroid: a dynamic ML-based android malware analyzer using system call traces’, *Cluster Comput*, vol. 23, no. 4, pp. 2789–2808, Sep. 2020, doi: 10.1007/s10586-019-03045-6.

[13] S. Shakya and M. Dave, ‘Analysis, Detection, and Classification of Android Malware using System Calls’, Aug. 2022, doi: 10.48550/arxiv.2208.06130.

[14] S. Lee, W. Jung, S. Kim, and E. T. Kim, ‘Android Malware Similarity Clustering using Method based Opcode Sequence and Jaccard Index’, *ICTC 2019 - 10th International Conference on ICT Convergence: ICT Convergence Leading the Autonomous Future*, pp. 178–183, Oct. 2019, doi: 10.1109/ICTC46691.2019.8939894.

[15] A. Lekssays, B. Falah, and S. Abufardeh, ‘A Novel Approach for Android Malware Detection and Classification using Convolutional Neural Networks’, *ICSOFT 2020 - Proceedings of the 15th International Conference on Software Technologies*, pp. 606–614, 2020, doi: 10.5220/0009822906060614.

[16] M. Farrokhmanesh and A. Hamzeh, ‘Music classification as a new approach for malware detection’, *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 2, pp. 77–96, Jun. 2019, doi: 10.1007/S11416-018-0321-2/FIGURES/8.

[17] F. Mercaldo and A. Santone, ‘Audio signal processing for Android malware detection and family identification’, *Journal of Computer Virology and Hacking Techniques*, vol. 17, no. 2, pp. 139–152, Jun. 2021, doi: 10.1007/S11416-020-00376-6/TABLES/3.

[18] R. Casolare, G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, ‘Mobile Family Detection through Audio Signals Classification’, *18th International Conference on Security and Cryptography (SECRYPT 2021)*, pp. 479–486, 2021, doi: 10.5220/0010543504790486.

[19] X. Zhao and D. Wang, ‘Analyzing noise robustness of MFCC and GFCC features in speaker identification’, *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 7204–7208, Oct. 2013, doi: 10.1109/ICASSP.2013.6639061.

[20] C. Kumar, F. Ur Rehman, S. Kumar, A. Mehmood, and G. Shabir, ‘Analysis of MFCC and BFCC in a speaker identification system’, *2018 International Conference on Computing, Mathematics and Engineering Technologies: Invent, Innovate and Integrate for Socioeconomic Development, iCoMET 2018 - Proceedings*, vol. 2018-January, pp. 1–5, Apr. 2018, doi: 10.1109/ICOMET.2018.8346330.

[21] Y. Zouhir and K. Ouni, ‘Feature extraction method for improving speech recognition in noisy environments’, *Journal of Computer Science*, vol. 12, no. 2, pp. 56–61, 2016, doi: 10.3844/JCSSP.2016.56.61.

[22] T. Irino and R. D. Patterson, ‘A time-domain, level-dependent auditory filter: The gammachirp’, *Citation: The Journal of the Acoustical Society of America*, vol. 101, p. 412, 1997, doi: 10.1121/1.417975.

[23] T. Irino and R. D. Patterson, ‘A dynamic compressive gammachirp auditory filterbank’, *IEEE Trans Audio Speech Lang Process*, vol. 14, no. 6, pp. 2222–2232, Nov. 2006, doi: 10.1109/TASL.2006.874669.

[24] Y. Zouhir and K. Ouni, ‘A bio-inspired feature extraction for robust speech recognition’, *Springerplus*, vol. 3, no. 1, pp. 1–8, Nov. 2014, doi: 10.1186/2193-1801-3-651/TABLES/5.

[25] A. ben Abdallah and Z. Hajaiej, ‘Improved closed set text independent speaker identification system using Gammachirp Filterbank in noisy environments’, *2014 IEEE 11th International Multi-Conference on Systems, Signals and Devices, SSD 2014*, 2014, doi: 10.1109/SSD.2014.6808815.

[26] M. Bouchamekh, B. Bousseksou, and D. Berkani, ‘Gammachirp filterbank based speech analysis for speaker identification, Proceedings of the 8th WSEAS International Conference on Computational intelligence, man-machine systems and cybernetics’, 2009. Accessed: Aug. 21, 2022. [Online]. Available: https://dl.acm.org/doi/10.5555/1736097.1736101

[27] Z. Hajaiej, K. Ouni, and N. Ellouze, ‘Gammachirp Filter Frond-End for Automatic Speech Recognition, SETIT 2007 4 th International Conference: Sciences of Electronic, Technologies of Information and Telecommunications-TUNISIA’, 2007.

[28] K. Salhi, Z. Hajaiej, and N. Ellouze, ‘A novel approach for auditory spectrum enhancement to improve speech recognition’s robustness’, *12th International Multi-Conference on Systems, Signals and Devices, SSD 2015*, Dec. 2015, doi: 10.1109/SSD.2015.7348134.

[29] A. Borisyuk, ‘Physiology and mathematical modeling of the auditory system’, *Lecture Notes in Mathematics*, vol. 1860, pp. 107–168, Jan. 2005, doi: 10.1007/978-3-540-31544-5_4.

[30] P. Tarwireyi, A. Terzoli, and M. O. Adigun, ‘BarkDroid: Android Malware Detection Using Bark Frequency Cepstral Coefficients’, *Indonesian Journal of Information Systems*, vol. 5, no. 1, pp. 48–63, Aug. 2022, doi: 10.24002/IJIS.V5I1.6266.

[31] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, ‘Very deep convolutional neural networks for raw waveforms’, *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 421–425, Jun. 2017, doi: 10.1109/ICASSP.2017.7952190.

[32] T. Kim, J. Lee, and J. Nam, ‘Sample-Level CNN Architectures for Music Auto-Tagging Using Raw Waveforms’, *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2018-April, pp. 366–370, Sep. 2018, doi: 10.1109/ICASSP.2018.8462046.

[33] L. Deng and Y. Gao, ‘Gammachirp filter banks applied in roust speaker recognition based on GMM-UBM classifier’, *International Arab Journal of Information Technology*, vol. 17, no. 2, pp. 170–177, Mar. 2020, doi: 10.34028/IAJIT/17/2/4.

[34] M. Jeevan, A. Dhingra, M. Hanmandlu, and B. K. Panigrahi, ‘Robust speaker verification using GFCC based i-vectors’, *Lecture Notes in Electrical Engineering*, vol. 395, pp. 85–91, 2017, doi: 10.1007/978-81-322-3592-7_9.

[35] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, ‘Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification’, *Proceedings - International Carnahan Conference on Security Technology*, vol. 2018-October, Dec. 2018, doi: 10.1109/CCST.2018.8585560.

[36] S. Mahdavifar, A. F. Abdul Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, ‘Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning’, *Proceedings - IEEE 18th International Conference on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress*, pp. 515–522, Aug. 2020, doi: 10.1109/DASC-PICOM-CBDCOM-CYBERSCITECH49142.2020.00094.

[37] S. Mahdavifar, D. Alhadidi, and A. A. Ghorbani, ‘Effective and Efficient Hybrid Android Malware Classification Using Pseudo-Label Stacked Auto-Encoder’, *Journal of Network and Systems Management*, vol. 30, no. 1, Jan. 2021, doi: 10.1007/S10922-021-09634-4.

Authors Profile

Paul Tarwireyi is a Lecturer in the Computer Science at the University of Zululand. He obtained his MSc degree Computer Science from the University of Fort Hare in 2008; His research interests are in cyber security, computer networks, cloud computing and machine learning. He is a member of IITPSA and SAICSIT. Email: tarwireyip@unizulu.ac.za

Alfredo Terzoli is currently a Professor of Computer Science at the University of Zululand. He obtained a Laurea cum Laude in Physics from the University of Pavia, Italy, in 1980 and moved into computing soon afterwards, working in the private sector for a while. He then moved to Rhodes University, South Africa, attracted by the possibility, existing there, to mix Computer Music and Artificial Intelligence, as well as to experience life in Africa. His research interests include real-time multimedia, telco-internet convergence, ICT in Education and ICT for Development. Email: terzolia@unizulu.ac.za

Matthew O Adigun is currently a Senior Professor of Computer Science at the University of Zululand. He obtained his doctorate degree in 1989 from Obafemi Awolowo University, Nigeria; A very active researcher in Software Engineering of the Wireless Internet, he has published widely in the specialised areas of reusability, software product lines, and the engineering of on-demand grid computing-based applications in Mobile Computing, Mobile Internet and ad hoc Mobile Clouds. His current research interests are in Software Engineering, Mobile Edge Computing Systems, Web intelligence, ICT4D, SDN, Machine learning and Security of Grid/Cloud/Fog Systems. He is a member of IEEE and SAICSIT and a reviewer of many journals. Email: adigunm@unizulu.ac.za