**Comparison Of MFCC, BFCC, And NGCC In Android Malware Detection**

Lungelo Mkhize and Paul Tarwireyi

Department of Computer Science

University of Zululand

South Africa

# **Abstract**

**The Android operating system has always been in the spotlight. Cybercriminals have recognized attracting the growing challenges. However, many researchers have achieved success by employing machine/deep learning techniques to build malware detection models based on popular Drebin malware datasets. The paper presents an audio feature malware detection system, using machine learning classifiers for the Drebin dataset (APK) which is treated as acoustic signals and we used audio feature extraction techniques named Normalized Gammachirp Cepstral Coefficients (NGCC), Bark Frequency cepstral coefficients (BFCC) and Mel-frequency cepstral coefficients (MFCC). Based on the results obtained from the experiment, BFCC is the most effective audio feature compared to other authors’ works. The BFCC audio feature has a high accuracy score of 99.0% with a precision of 99.0%, F1-Score of 99.0%, and recall of 99.0% compared to other authors although NGCC and MFCC results are encouraging but (CHENG LIN LI et al., 2019) has the better results than them.**

**Keywords***: Android malware detection, Mel Frequency cepstral Coefficients, Bark frequency cepstral coefficients, and Normalized Gammachirp Cepstral Coefficients.*

# **INTRODUCTION**

Android is the operating system that has gained significant popularity in the mobile telecommunications industry due to its large user base.According to 2020 statistical data, there are approximately 3.5 billion mobile device users around the entire world (Conker, 2020). The increase in the development of android devices has become the major attack target of malware. Malware can be a malicious file or code deployed to the system to perform the task the attacker intended. Since there are variants of malware, there are a lot of ways to deploy android malware into the system. These ways include downloading malicious apps, opening suspicious emails, receiving text messages/voicemail phishing, etc. (Arslan RS,2021).

Several alternative ways for detecting malware on Android have been offered by researchers. The techniques of malware detection that have been employed may be divided into three groups: static, dynamic, and hybrid analysis based. Although it is challenging to evaluate binary code due to the proliferation of obfuscation techniques, such as encryption and data masking, etc., the static analysis looks at the binary code and attempts to identify any code that has not yet been executed. Since static analysis relies on a predefined signature database, it is unable to identify fresh, undiscovered malware. Malware may be executed, and its activities tracked using the dynamic analysis approach, which has proved successful in detecting malware. Both static and dynamic approaches detect malware by comparing unusual and predictable behaviour.

More than 50 million instances of malware and potentially unwanted applications (PUA) for Android have been discovered, according to data (AV-TEST institute, 2021).

Chart, bar chart

Description automatically generated

Figure 1: Number of Android malware per year (AV-TEST Institute, 2021)

Smartphones have become a crucial part of our everyday lives; we almost exclusively rely on them for the critical data we save on them. However, the flexibility of android smartphones has allowed attackers to take advantage of them. Hackers can install their malicious applications onto android phones since android devices enable third parties to install the program into their system. When an application is installed, the user is prompted to approve all permissions. Cybercriminals typically used this as an advantage to request the user's full consent before accessing confidential data. When installing the application, the user consents to all permissions without realizing that they are being requested, necessitating the necessity to identify Android malware before it infiltrates the device.

Many researchers have examined the effectiveness of the models in the Drebin dataset by applying machine-learning techniques for the identification of android malware. In this paper, we used the same dataset to develop the audio feature malware detection system and the dataset is treated in the format of WAV. We applied audio feature extraction techniques, named Bark Frequency cepstral coefficient (BFCC), Normalized Gammachirp Cepstral Coefficients (NGCC), and Mel Frequency Cepstral Coefficients (MFCC**).** When users use the ASR system on their smartphones to retrieve information while being interrupted by other people's conversations in a vehicle or subway system (Ta-Wen Kuan, 2015), for example, the accuracy performance of ASR is evaluated using the BFCC approach. However, although MFCC has been investigated for its performance in speech recognition systems, the Fourier transform and the triangular Mel-filter bank used in MFCC is not representative of the sound wave sensitivity at the basilar membrane in the human auditory system, as well as having lower robustness in the presence of additive noise (Ta-Wen Kuan, 2015).

We tested our model using two datasets of Android malware: DREBIN (Daniel Arp et al, 2014) and the Google Play Store dataset that was directly downloaded, which each included 5,560 samples (APK). We used the key metrics to evaluate performance: accuracy, false positive rate (FPR), precision, recall, F1, and training time.

# **RELATED WORK**

After a thorough review of the research in the field of Android malware detection. Android malware has become a security threat on mobile devices. I saw a variety of study objectives. Many research papers look at present solutions to dealing with the malware detection problem. Various methods and tactics are reviewed to identify android malware. (Pallavi. K. et al.2020) employed the approach to identify malware in Android applications using a machine learning classifier that incorporated both static and dynamic features. They also extract the features using dynamic and static methodologies. Using the Environment for knowledge analysis to train machine learning classifiers and obtain the results using K-fold cross-validation.

The strategy is put into practice using both static and dynamic analytical techniques. Static analysis is used to extract the binary code from the executable file and prepare the function set. As a result of the execution time analysis, the behavior of executable files is examined. Each approach has its significance, and each technique has certain benefits and drawbacks for identifying malware and other security holes. A technique of identifying malware utilizing static and dynamic properties retrieved from the executable file was proposed by (Awan, 2017). Based on static malware analysis, the authors presented their results. According to their approach, the authors employed import functions, grayscale image methods, and OPODODE NGR image techniques for the extraction procedure (Zhongzhi S., 2021). They come up with a powerful technique for spotting malware in Windows operating systems. Extraction of the executable file's many features. These characteristics are utilized as inputs for several machine-based classifiers that categorize malware executable files (Zhongzhi Shi, 2021).

(Mohamed S.A, 2022) other studies investigate the performance of four machine learning classifiers that can find malware depending on dynamic and static features. (Long Wen et al. 2017) developed an algorithm to detect android malware called SVM, which is different from the traditional detection method. The proposed method shows a higher and lower error detection rate than the traditional approach. Dynamic tools like Scandroid, DroidRange, and vetdroid can identify malicious activities. Tools based on static analysis like Androgaurd, Pscout, FlowDroid, etc. Can detect and prevent the malicious application from being installed on the device, (Sangeeta R et al., 2019). Various data mining models are trained, and performance measures like accuracy and recall are assessed and compared (Mohamed S.,2022).

Dynamic analysis approaches, as opposed to static analysis, govern the malware execution process. At runtime, collect, observe, and record malware behavior characteristics. The dynamic analysis approach is often executed in a secure virtual environment known as a sandbox. Cuckoo Sandbox (Cuckoo Sandbox, 2019) and CW Sandbox are two popular dynamic analysis sandboxes.

(CWSandbox, 2019) The primary purpose of the sandbox is to detect dangerous activity in malware and prevent it from harming the host system. Both static and dynamic analysis methodologies have benefits and drawbacks. The major benefit of static analysis over dynamic analysis is that it does not have the overhead cost of executing the program. Static analysis approaches, on the other hand, have limitations due to a lack of support for packaging and complicated obfuscated code. Dynamic analysis, as opposed to static analysis, can successfully evaluate bundled and disguised malware. This is because the virus must unpack itself while it is executing. As a result, the original and malicious code will be placed in the main memory. However, the fundamental disadvantage of dynamic analysis is that it takes time and resources. Malwaresamples must be evaluated independently, limiting the use of dynamic analysis in commercial applications.

For machine learning algorithms to be as successful as required (Fallah et al.,2019) benchmarking must occur first. The discovery of the specific family of harmful apps forms the basis of this method. The authors also demonstrate how combining techniques is essential to get consistent outcomes across platforms based on selected datasets or malware classifications. The authors suggest using network-based detection methods and machine learning in this circumstance. Machine learning should be applied both to supervised and unsupervised approaches to produce useful results that can be used in the decision-making process. However, However, the study does not specifically show how the machine-learning algorithms will deal with new malware types that have not yet been tested using available techniques. Therefore, algorithms must be continually updated to collect and detect the latest malware families and enable machine-learning techniques.

(Md. Shohel R., 2021) makes use of the DREBIN dataset to evaluate classifiers for Android malware detection that use tree-based machine learning. From the Drebin dataset, they chose 11,120 applications for testing, of which 5,560 had malicious files and the rest were benign. The accuracy of the SVM classifiers was 94%, whereas the accuracy of the random forest classifiers was superior to both at 97.24%. The model proposed by (Yeima et al. 2021) that uses static analysis based on Bayesian classification to provide indicators of potentially malicious activities yielded the most effective results, with TPR (True Positive Rate) of 90.6%, FNR (False Negative Rate) of 0.094%, an accuracy of 93.5%, and AUC (Area Under Curve) of 97.22%. Five supervised machine learning algorithms were used in the dynamic analysis model proposed by (Feizollah et al., 2019), with KNN generating the highest results with a TPR of 99.94% and an FPR of (False Positive Rate) of 0.06%.

# **RESEARCH PROBLEM**

Our daily lives have become more and more dependent on android devices. This is because most people use them for learning and research, online banking, and finance, as well as voice, message, and email communication. The dominance of this operating system has made it the major target for cyber attackers. Cyber attackers deploy malware on android devices to delete, modify and steal critical information. However, Static and Dynamic malware detection approach has been introduced to solve this issue. However, they are not effective nowadays because cyber attackers have studied these strategies and developed updated variants of malware It has become difficult to develop new techniques to combat new malware. Although dynamic has executed and analyzed the malware threat and its behavior. In this study, we focus on audio-based features in malware detection.

1. **RESEARCH QUESTIONS**

In this study, we consider the following research questions to develop an audio feature malware detection system:

1. What is the state of the art regarding malware detection?
2. How can we design a malware detection system using audio features?
3. How do BFCC, MFCC, and NGCC compare in Android malware detection?

# **RESEARCH GOALS AND OBJECTIVES**

This study aims to compare BFCC, MFCC, and NGCC results to develop an audio malware detection system:

1. To determine the state of the art regarding malware detection.
2. To compare BFCC, MFCC, and NGCC features with other previous work in android malware detection.
3. To provide the most effective audio feature for malware detection.

# **RESEARCH METHOD**

1. *Proposed study*

This paper treats the Android application as an audio signal with unique properties that can be analyzed using Automated Signal Recognition techniques. Mel-frequency cepstral coefficients (MFCC), Bark Frequency cepstral coefficients (BFCC), and Normalized Gammachirp Cepstral Coefficients.

1. *Dataset description*

In this study, the dataset that was used in this study was collected from the Drebin website and google play store. DREBIN: Contains 5,560 malware files and 2201 benign APK files collected from August 2010 to October 2012, and the Google Play store contains only benign. This is one of the most popular datasets for Android malware detection. When evaluating the DREBIN dataset, we randomly sampled 5,600 clean files to match the number of malware samples in this dataset. Finally, the results of the machine learning algorithm tests on the two datasets are obtained and compared.

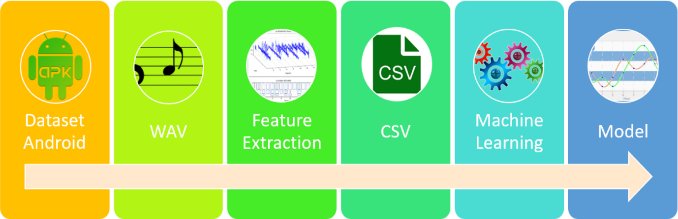
1. *Feature engineering*

In this process, we performed three audio feature extraction processes, which are Normalized Gammachirp Cepstral Coefficients (NGCC), Bark Frequency Cepstral Coefficients (BFCC), and Mel Frequency Cepstral Coefficients (MFCC).

* Drebin and Google Play Store datasets are collected and unzipped into different folders of these three audio signals.
* Both datasets have been cleansed to get rid of any duplicate and damaged APK files.
* To establish the distribution of the different samples in the dataset, data exploration, and validation are performed.

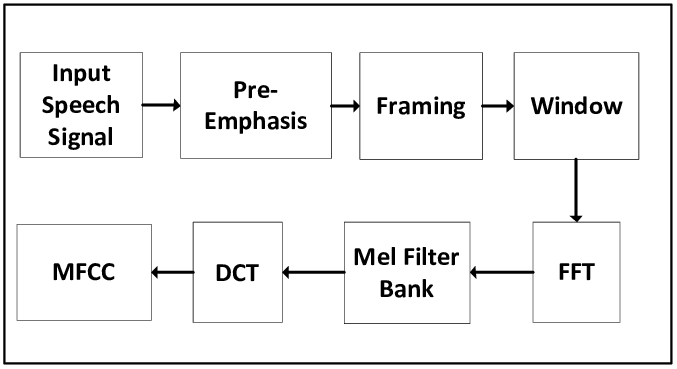
1. Data transformation

Drebin dataset comes with the raw Android Application Package (APK) that contains a variety of components such as classes, methods, libraries, etc. APK files were converted into WAV files. Audio feature extraction was generated to produce BFCC, MFCC, and NGCC CSV files. As the diagram is shown below.

*Figure 2: Proposed methodology architecture diagram (Mercaldo. F & Santone. A, 2021)*

1. Cepstral-based features
2. Mel Frequency Cepstral Coefficients (MFCC)

Windowing the signal, using the DFT, taking the log of the magnitude, warping the frequencies on a Mel scale, and then incorporating the inverse DCT are the main steps in the MFCC method of feature extraction. In this paper, we use this audio feature extraction technique.

 Figure 3: Flowchart of MFCC Feature Extraction (Magre S.B & Janse P.V, 2014)

1. Bark Frequency cepstral coefficient (BFCC)

BFCC defined as Bark Frequency Cepstrum, which is based on the linear cosine transform of a log spectrum on a non-linear Bark scale of frequency, is a short-time power spectrum representation of a signal. Can be calculated using these equations, are which the same as MFCC.





Equation 1: BFCC (Tarwireyi P., 2022)

1. Normalized Gammachirp Cepstral Coefficients( NGCC)

Equation 2:NGCC

Text

Description automatically generated

where M is the total number of NGCC coefficients, N is the number of auditory filterbank channels, and Log (Xk) is the energy output in the logarithmic scale of the kth filter (k = 1, 2..., N). N and M are set to be 34 and 12, respectively, for the NGCC calculations.

1. *Machine Learning Algorithms*

In this study, Jupyter notebook was used to develop Machine learning Algorithms to compare them with other existing literature to achieve the most effective audio feature for malware detection. The algorithms that were evaluated were Logistic Regression, Decision Tree, Random Forest, Cat boost, Extra Tree, Support Vector Machine, K-Nearest Neighbour, Naïve Bayes, and Bagging Decision Tree.

1. *Evaluation Metrics*

To compare machine learning algorithms, accuracy, F1-score, precision, recall, and AUC were considered. For evaluation metrics, the following equations were utilized.

# **RESULTS AND DISCUSSION**

The results of the proposed method are shown in this section. After the collection of datasets from the Drebin website containing 5,5600 malware samples and the Google Play Store containing 5,600 benign samples, we converted to the Wav. Audio format. Mel Frequency cepstral coefficients, Bark Frequency cepstral Coefficients, and Normalized Gammachirp Cepstral Coefficients were generated from audio files for analysis.

1. *Experimental setup*

The work was done on a desktop with an x64-based processor, an Intel® CoreTM i7-9700 CPU processor running at 3.00 GHz, and 16 GB of memory running Windows 11 Pro. Jupyter and Python 3.9 were used to create the code for the machine-learning algorithms. For model evaluation, 9 machine-learning classifiers were used in this experiment.

1. *Classification performance*

We present the tables that show the audio features models’ performance.

*Table 1: BFCC models performance*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Classifier used | Test Accuracy | Precision | Recall | F1 Score | AUC | Train time  (s) | Test time  (s) |
| Extra Tree | 0.9901 | 0.99 | 0.99 | 0.99 | 0.99 | 0.040 | 0.03948 |
| SVC | 0.9882 | 0.97 | 0.97 | 0.97 | 0.99 | 0.034 | 0.348 |
| Naïve Bayes | 0.9872 | 0.785 | 0.90 | 0.78 | 0.98 | 0.002 | 0.00173 |
| Random Forest | 0.9872 | 0.98 | 0.98 | 0.98 | 0.96 | 0.029 | 0.02835 |
| KNN | 0.9825 | 0.97 | 0.98 | 0.97 | 0.98 | 0.116 | 0.11547 |
| Cat boost | 0.9768 | 0.98 | 0.98 | 0.98 | 0.97 | 0.435 | 0.43388 |
| Decision Tree | 0.9688 | 0.98 | 0.98 | 0.98 | 0.96 | 0.002 | 0.00154 |
| Logistic Regression | 0.9217 | 0.92 | 0.92 | 0.92 | 0.92 | 0.002 | 0.00126 |
| Bagging DT | 0.7713 | 0.98 | 0.98 | 0.98 | 0.97 | 0.036 | 0.03520 |

Table 2: NGCC models performance

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Classifier used | Test Accuracy | Precision | Recall | F1 Score | AUC | Train time  (s) | Test time  (s) |
| KNN | 0.9471 | 0.94 | 0.94 | 0.94 | 0.95 | 0.152 | 0.15244 |
| SVC | 0.9458 | 0.94 | 0.94 | 0.94 | 0.74 | 0.456 | 0.45619 |
| Naïve Bayes | 0.9441 | 0.75 | 0.73 | 0.72 | 0.95 | 0.003 | 0.00272 |
| Random Forest | 0.9423 | 0.94 | 0.95 | 0.95 | 0.94 | 0.35 | 0.03462 |
| Extra Tree | 0.9379 | 0.95 | 0.95 | 0.95 | 0.94 | 0.044 | 0.04356 |
| Cat boost | 0.9296 | 0.95 | 0.95 | 0.94 | 0.93 | 0.055 | 0.05473 |
| Decision Tree | 0.9021 | 0.94 | 0.95 | 0.95 | 0.90 | 0.002 | 0.00212 |
| Logistic Regression | 0.8982 | 090 | 0.90 | 0.90 | 0.90 | 0.002 | 0.00168 |
| Bagging DT | 0.7502 | 0.95 | 0.95 | 0.95 | 0.76 | 0.038 | 0.03779 |

Table 3: MFCC models performance

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Classifier used | Test Accuracy | Precision | Recall | F1 Score | AUC | Train time  (s) | Test time  (s) |
| Naïve Bayes | 0.8747 | 0.61 | 0.54 | 0.47 | 0.88 | 0.003 | 0.00260 |
| Random Forest | 0.8730 | 0.87 | 0.88 | 0.87 | 0.88 | 0.036 | 0.03646 |
| KNN | 0.8704 | 0.85 | 0.86 | 0.86 | 0.88 | 0.129 | 0.12913 |
| SVC | 0.8590 | 0.85 | 0.86 | 0.85 | 0.86 | 0.911 | 0.91078 |
| Extra Tree | 0.8555 | 0.87 | 0.89 | 0.88 | 0.86 | 0.048 | 0.04803 |
| Cat boost | 0.8516 | 0.87 | 0.88 | 0.87 | 0.86 | 0.105 | 0.10550 |
| Logistic Regression | 0.8228 | 0.82 | 0.82 | 0.82 | 0.82 | 0.002 | 0.00163 |
| Decision Tree | 0.8180 | 0.82 | 0.82 | 0.82 | 0.81 | 0.002 | 0.00221 |
| Bagging DT | 0.6095 | 0.82 | 0.83 | 0.82 | 0.54 | 0.038 | 0.03818 |

1. *Benchmarking with other works.*

In this section, we compare the results of this work with other authors ‘work using the same dataset as this study, with the purpose of achieving the most effective audio feature that can be used in the future as an audio feature technique to detect malware on the android operating system.

Table 4: Comparison

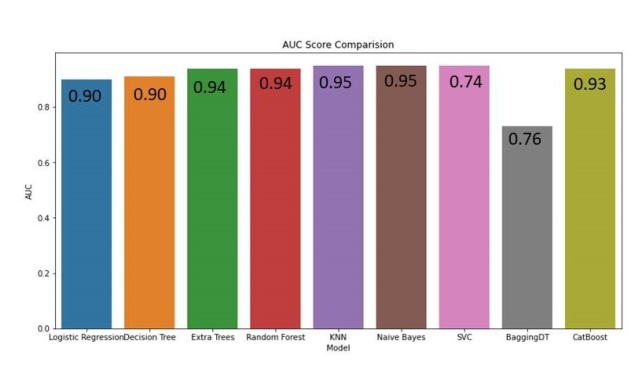
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Authors | Models | Accuracy | Precision | Recall | F1-Score |
| Rahul Y, 2022 | Random Forest | 0.981 | 0.943 | 0.860 | 0.900 |
| Qing Wu, 2020 | K-Nearest Neighbour | 0.976 | 0.969 | 0.874 | 0.918 |
| CHENGLIN Li, et al. 2022 | Naïve Bayes | 0.977 | 0.962 | 0.992 | 0.977 |
| Mohamed S.A, 2022 | Random Forest | 0.973 | 0.973 | 0.785 | 0.869 |
| Md Shohel R, 2018 | Random Forest | 0.938 | 0.950 | 0.930 | 0.950 |
| This study  (BFCC) | Extra Tree | 0.990 | 0.99 | 0.99 | 0.990 |
| This study (NGCC) | KNN | 0.947 | 0.94 | 0.94 | 0.94 |
| This study (MFCC) | Naïve Bayes | 0.874 | 0.61 | 0.54 | 0.47 |

**Interpretation.**

Based on the results, BFCC has a high accuracy score of 0.99, a precision of 0.99, an F1-Score of 0.99, and a recall of 0.99. NGCC and MFCC have the most effective results, but (Rahul Y., 2022) have better results than them. In other words, the BFCC feature is most likely to detect malicious applications on Android.

1. *Visualization metrics*

Below are the graphs of the Area under the curve and the accuracy for different feature extraction techniques. The area under the curve shows how much the model is capable of distinguishing between classes. It also shows the accuracy is used to identify which model is most effective at detecting Android malware.



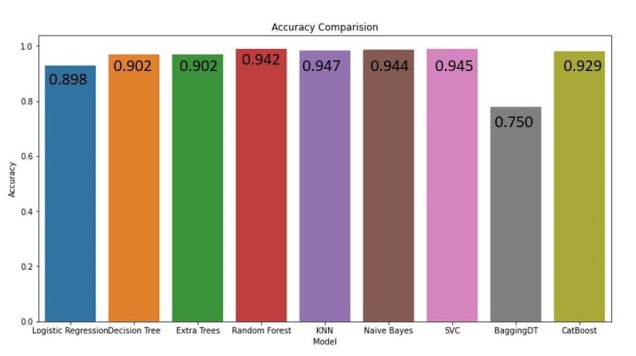
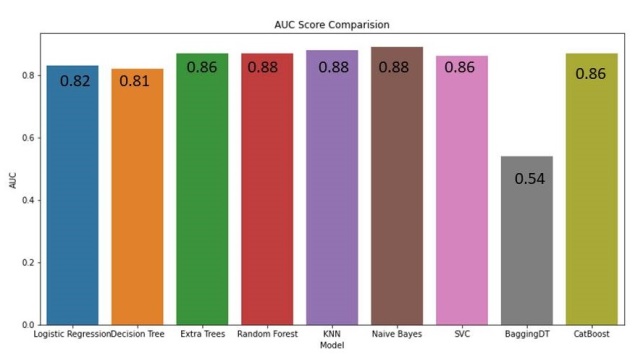


Figure 4: NGCC graphs for accuracy and AUC



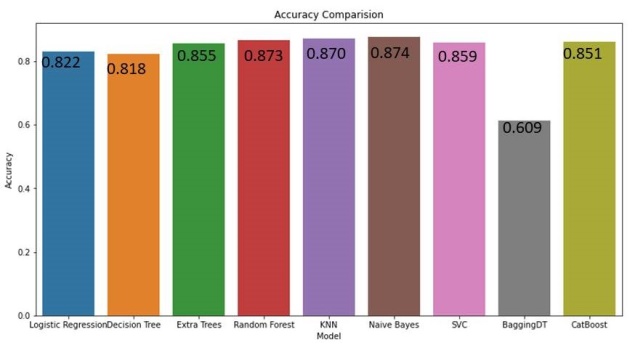
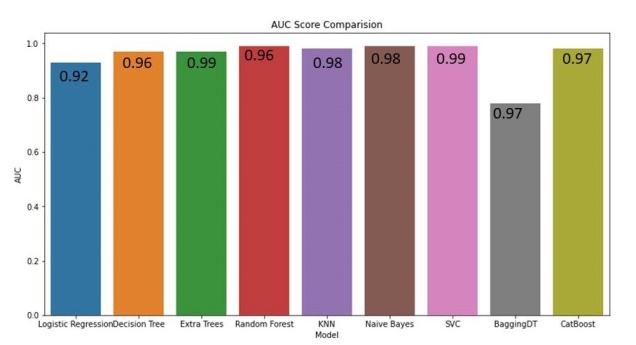


Figure 5: MFCC graphs for accuracy and AUC



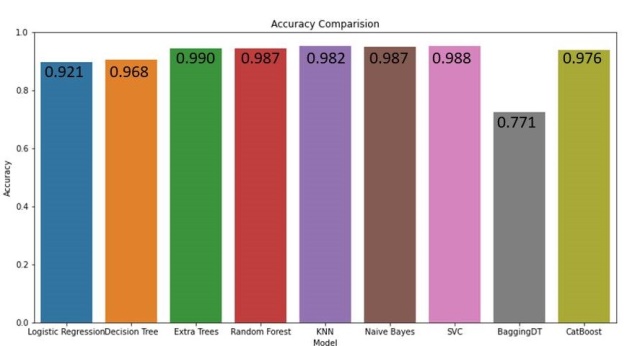


Figure 6: BFCC for Accuracy and AUC

**Interpretation**.

The audio features graphs for accuracy and the area under the curve are shown. On BFCC the algorithm that has the highest accuracy is Extra Tree with 99.0%, with the area under the curve of 99.0%. The NGCC feature obtained an accuracy of 95.0% with the area under the curve of 95.0%. The last feature is MFCC with an accuracy of 86.0% and an area under the curve of 88.0%. Meaning the BFCC feature is the one that can detect malware on android application than other features.

1. Precision-recall

Since accuracy alone is insufficient to comprehend the performance of classification models, precision and recall metrics are employed to evaluate machine learning algorithms. In this study, we consider the best precision-recall graph for each audio feature.

Graphical user interface

Description automatically generated with low confidence

Figure 7: BFCC precision-recall

Shape

Description automatically generated with medium confidence

Figure 8: MFCC precision-recall

A picture containing shape

Description automatically generated

Figure 9: NGCC precision-recall

**Interpretation**.

The BFCC feature with the Extra tree algorithm is probably the most effective. This is due to its precision score of 99.0%, meaning that it has the most accurate ratio of malicious apps that are classified correctly. Audio features (BFCC, MFCC, and NGCC) have algorithms that provide high precision and recall rates, which means they return a high number of accurately labelled results.

# **CONCLUSION**

In this paper, nine machine-learning algorithms were used to evaluate the performance of the proposed method for Android malware detection. As the above results indicate, the most accurate classifier for the prediction of android malware is Extra Trees for Bark Frequency Cepstral Coefficients which achieved the highest score of 99.0%. After the comparison of the previous studies with this study, we find that the Bark Frequency Cepstral Coefficients is the most effective among other Authors, which means that audio features can be the solution in malware detection.

The contribution of this paper to Android malware detection is that we used the audio feature to detect malware on the android operating system and we treat the APK dataset from Drebin as signal modelled to perform audio feature extraction techniques. Which is the first thing in this dataset to use such a format. For the future of android malware, More research are required to determine which feature is the most effective in audio-based features.

1. **R****EFERENCES**

[1] Ashawa, M. and Morris, S. (2019a) ‘Analysis of Android Malware Detection Techniques: A Systematic Review’, *International Journal of Cyber-Security and Digital Forensics*, 8(3), pp. 177–187. Available at: https://doi.org/10.17781/p002605.

[2] Kuan, T.-W. *et al.* (2016) ‘A robust BFCC feature extraction for ASR system’, *Artificial Intelligence Research*, 5(2). Available at: https://doi.org/10.5430/air.v5n2p14.

[3] Li, C. *et al.* (2019) ‘Android Malware Detection Based on Factorization Machine’, *IEEE Access*, 7, pp. 184008–184019. Available at: https://doi.org/10.1109/access.2019.2958927.

[4] Lindorfer, M., Neugschwandtner, M. and Platzer, C. (2015) ‘MARVIN: Efficient and Comprehensive Mobile App Classification through Static and Dynamic Analysis’, *2015 IEEE 39th Annual Computer Software and Applications Conference* [Preprint]. Available at: https://doi.org/10.1109/compsac.2015.103.

[5] Mahindru, A. and Sangal, A.L. (2020) ‘MLDroid—framework for Android malware detection using machine learning techniques’, *Neural Computing and Applications*, 33(10), pp. 5183–5240. Available at: https://doi.org/10.1007/s00521-020-05309-4.

[6] Mahindru, A. and Singh, P. (2017) ‘Dynamic Permissions based Android Malware Detection using Machine Learning Techniques’, *Proceedings of the 10th Innovations in Software Engineering Conference* [Preprint]. Available at: https://doi.org/10.1145/3021460.3021485.

[7] Mishra, J. *et al.* (2021) ‘Duplicates in the Drebin Dataset and Reduction in the Accuracy of the Malware Detection Models’, *2021 26th IEEE Asia-Pacific Conference on Communications (APCC)* [Preprint]. Available at: https://doi.org/10.1109/apcc49754.2021.9609892.

[8] Odusami, M. *et al.* (2018) ‘Android Malware Detection: A Survey’, *Communications in Computer and Information Science*, pp. 255–266. Available at: https://doi.org/10.1007/978-3-030-01535-0\_19.

[9] Pomianos, G. *et al.* (2003a) ‘Recent advances in the automatic recognition of audiovisual speech’, *Proceedings of the IEEE*, 91(9), pp. 1306–1326. Available at: https://doi.org/10.1109/jproc.2003.817150.

[10] Rana, Md.S., Rahman, S.S.M.M. and Sung, A.H. (2018) ‘Evaluation of Tree Based Machine Learning Classifiers for Android Malware Detection’, *Computational Collective Intelligence*, pp. 377–385. Available at: https://doi.org/10.1007/978-3-319-98446-9\_35.

[11] Sulapas Mapa, J. (2018) ‘Android Untrusted Detection with Permission Based Scoring Analysis’, *International Journal of Information Technology Convergence and Services*, 8(3/4), pp. 01–12. Available at: https://doi.org/10.5121/ijitcs.2018.8401.

[12] Ul  Hassan, C.A., Khan, M.S. and Shah, M.A. (2018c) ‘Comparison of Machine Learning Algorithms in Data classification’, *2018 24th International Conference on Automation and Computing (ICAC)* [Preprint]. Available at: https://doi.org/10.23919/iconac.2018.8748995.

[13] Wen, L. and Yu, H. (2017) ‘An Android malware detection system based on machine learning’, *AIP Conference Proceedings* [Preprint]. Available at: https://doi.org/10.1063/1.4992953.

[15] Wu, Q. *et al.* (2020) ‘MVIIDroid: A Multiple View Information Integration Approach for Android Malware Detection and Family Identification’, *IEEE MultiMedia*, 27(4), pp. 48–57. Available at: https://doi.org/10.1109/mmul.2020.3022702.

[16] Youchao Dong (2017) ‘Android Malware Prediction by Permission Analysis and Data Mining’, *University of Michigan - Deep Blue* [Preprint]. Available at: http://deepblue.lib.umich.edu/bitstream/2027.42/136197/1/YouchaoDong\_Thesis\_0327.pdf.

[17] Yuan, B. *et al.* (2020) ‘Byte-level malware classification based on markov images and deep learning’, *Computers &Amp; Security*, 92, p. 101740. Available at: https://doi.org/10.1016/j.cose.2020.101740.

[18] Yumlembam, R. *et al.* (2022) ‘IoT-based Android Malware Detection Using Graph Neural Network With Adversarial Defense’, *IEEE Internet of Things Journal*, pp. 1–1. Available at: https://doi.org/10.1109/jiot.2022.3188583.

[19] Zouhir, Y. and Ouni, K. (2016) ‘Feature Extraction Method for Improving Speech Recognition in Noisy Environments’, *Journal of Computer Science*, 12(2), pp. 56–61. Available at: https://doi.org/10.3844/jcssp.2016.56.61.

[20] Zwane, S., Tarwireyi, P. and Adigun, M. (2019) ‘Ensemble Learning Approach for Flow-based Intrusion Detection System’, *2019 IEEE AFRICON* [Preprint]. Available at: https://doi.org/10.1109/africon46755.2019.9133979.