**Indonesian Journal of Electrical Engineering and Computer Science**
Vol. 35, No. 2, August 2024, pp. 1244~1252
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v35.i2.pp1244-1252


# Android malware detection using GIST based machine learning and deep learning techniques

**Ponnuswamy Udayakumar[1], Srilatha Yalamati[2], Lavadiya Mohan[3], Mohd Junedul Haque[4], Gaurav Narkhede[5], Krishna Mohan Bhashyam[6]**
[1]Department of Artificial Intelligence and Data Science, Akshya College of Engineering and Technology, Kinathukadavu, India
[2]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vijayawada, India
[3]Department of CSE (Internet of Things), Balaji Institute of Technology and Science (Autonomous), Narsampeta, India
[4]School of Computer Sciences and Engineering, Sandip University, Nashik, India
[5]Department of Electrical and Electronics Engineering, Dr Vishwanath Karad MIT World Peace University, Pune, India
[6]Department of Information Technology, RVR and JC College of Engineering, Guntur, India

**Article Info**

***Article history:***

Received Mar 3, 2024
Revised Apr 9, 2024
Accepted Apr 13, 2024

***Keywords:***

Android
Deep learning
GIST
Machine learning
Malware detection

**ABSTRACT**

In today's digital world, Android phones play a vital part in a variety of facets of both professionals and individuals' personal and professional lives. Android phones are great for getting things done faster and more organized. The proportionate increase in the number of malicious applications has also been seen to be expanding. Since the play store offers millions of apps, detection of malware apps is challenging task. In this paper, a methodology is introduced for detecting malware in Android applications through the utilization of global image shape transform (GIST) features extracted from grayscale images of the applications. The dataset comprises samples of both malware and benign apps collected from the virus share website. After converting the apps into grayscale images, GIST features are extracted to capture their global spatial layout. Various machine learning (ML) algorithms, such as logistic regression (LR), k-nearest neighbour (KNN), AdaBoost, decision tree (DT), Naïve Bayes (NB), random forest (RF), support vector machine (SVM), extra tree classifier (ETC), and gradient boosting (GB), are employed to classify the applications according to their GIST features. Furthermore, a feed forward neural network (FFNN) is utilized as a deep learning (DL) technique to further improve the accuracy of classification. The performance of each algorithm is evaluated using metrics such as accuracy, precision and recall. The results demonstrated that the FFNN achieves superior accuracy compared to traditional ML classifiers, indicating its effectiveness in detecting malware in Android apps.



***Corresponding Author:***

Srilatha Yalamati
Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation
Vijayawada, Andhra Pradesh, India
Email: srilatha.yalamati@gmail.com

## 1. INTRODUCTION

The proliferation of mobile devices, particularly those running the Android operating system (OS), has led to a surge in the development and distribution of mobile applications. While this has brought about immense convenience and innovation, it has also opened the door to various security threats, including the proliferation of malware targeting Android apps. Malicious software, or malware, poses significant risks to users, ranging from data theft and financial fraud to device compromise and privacy breaches.

To address the growing concern of malware in Android apps, extensive research efforts have been devoted to developing effective detection methods. Traditional approaches typically rely on static and dynamic analysis techniques, which often require access to the app's executable code or runtime behavior. However, these methods may be limited by their dependence on specific app features or behaviors, making them susceptible to evasion tactics employed by sophisticated malware variants.

In this work, a novel method was proposed for detecting malware in Android apps by utilizing global image shape transform (GIST) features extracted from grayscale images of the app's user interface. The approach offered several advantages over traditional techniques, including the ability to analyze apps without requiring access to their executable code or runtime behavior. By representing apps as grayscale images and extracting GIST features to capture their global spatial layout, the aim was to provide a robust and versatile framework for malware detection.

Roy *et al.* [1] presented an Android malware detection technique using supervised learning. The method detected malicious application programming interface (API) calls and unusual behaviors, offering insights for researchers and users while suggesting avenues for future Android system technologies. Chowdhury *et al.* [2] highlighted the rising threat of Android malware to mobile device security and data integrity. Machine learning (ML) approaches for Android malware detection were tested, discussing security and malware issues on the platform. Later, they explored supervised, unsupervised, and ML detection methods and compared their effectiveness, assessing metrics. The evaluation revealed method flaws and suggested further research. It provided a detailed overview of Android malware detection using ML and its historical context. Awais *et al.* [3] used ANTI-ANT method to identify and prevent mobile malware. They targeted Botnets, Rootkits, SMS malware, Spywares, app installers, and ransomware. Three detection layers application, user background, and package formed the foundation. The extraction and categorization of features employed static and dynamic studies. One-shot learning-based Siamese neural networks were developed to recognize and categorize malware attacks [4]. It tested the strategy on 9,470 benign and 5,550 malware apps from Drebin. Several steps like pre-processing, data splitting, model architecture, training, and assessment done and reported good results.

A static feature-based ML model for Android malware detection was presented in [5]. It extracted features from a fresh dataset of co-existing permissions and API requests at different combination levels using the FP-growth method. The model was accurate using multiple ML techniques, including random forest (RF) employing permissions features at the second combination level. Alamro *et al.* [6] introduced the ensemble technique for automated Android malware detection using an optimal algorithm approach. They employed data preprocessing and an ensemble of three ML models, support vector machine (SVM), KELM, and neural networks. Parameter tuning was done resulting in improved detection. Rule-based and ML-based specific-type detectors were used to detect Android malware before and after installation [7]. It was non-invasive and obtained application functionalities without breaching licensing. Experiments on an Android smartphone showed the solution was three times quicker and used ten times less CPU, saving energy. It had much greater balanced accuracy, nine times less false positives, and ten times fewer false negatives than state-of-the-art systems. Banik and Singh [8] introduced a novel Android malware detection method using genuine Android permissions. They reverse-engineered APKs to extract genuine permission features and utilized frequent pattern growth to identify common permission combinations. These pairs were then inputted into a multi-layered neural network model and five traditional ML models for comparison. Evaluation metrics showed over 96% accuracy on both the Drebin dataset and a custom dataset from the past five years.

Mahindru *et al.* [9] introduced "YarowskyDroid," a technique for identifying malware-infected apps using semi-supervised ML and federated learning. Apps were installed locally on users' smartphones for privacy protection. Information from users improved the malware detection algorithm. The study addressed users' inability to detect malware in downloaded apps by proposing a semi-supervised learning technique. Lakshmanarao and Shashi [10] extracted opcodes from Android apps and applied recurrent neural network (RNN) for malware detection. The long short-term memory (LSTM) variant of RNN used in the experiment and reported good detection rate. Subash *et al.* [11] used static permissions and ML to identify Android malware. They performed API analysis on 400 Android apps to identify malicious activity. Later, trained and compared three ML algorithms after preprocessing. Sharma and Babbar [12] detected internet of things (IoT) Android malware using ML. This approach builds an ML model from Android malware samples and excellent applications. IoT malware detection uses ML methods including Naïve Bayes (NB), k-nearest neighbour (KNN), decision tree (DT), and RF on the Android Malware dataset and reported good results. Shatnawi *et al.* [13] proposed a static base classification technique for malware detection based on Android permissions and API calls to strengthen malware detection efforts. A large new Android malware dataset (CICInvesAndMal2019) was used with three popular ML algorithms: SVM, KNN, and NB.

Droos *et al.* [14] proposed ML classifiers for malware detection. To maximize detection accuracy, the algorithm used a feature set from the CICMalDroid2020 dataset to classify each APK as malicious or legal. Results showed that RF was the most accurate ML classifier. Bandi and Sherpa [15] detected

CICAndMal2017 Android malware using ML. Feature engineering was used to find the most important characteristics from a balanced dataset extracted by random sampling. The balanced dataset with specified characteristics trains ML algorithms. All models were first trained using 'Label' and subsequently 'Family'. Both examples used RF to get 99% accuracy. Alkahtani and Aldhyani [16] proposed ML techniques for malware detection. Several ML and DL algorithms applied and achieved good detection rate. Lakshmanarao and Shashi [17] applied ensemble learning for malware detection. Two types of ensembles stacking and blending applied and achieved accuracy of more than 95%. Kanchhal and Murugaanandam [18] built and injected Android malware onto an Android device or emulator, hiding it from the victim. The victim system provided vital data. Additionally, RF, ML discovered the virus. Alani and Awad [19] developed an explainable ML based lightweight Android malware detection method. To distinguish harmful and benign malware, the suggested approach used application characteristics. Over 98% accuracy was achieved with a tiny device footprint in testing. It was also described through shapley additive explanation (SHAP) values.

Ban *et al.* [20] explored how string properties like malware's security-sensitive APIs affected the deep learning (DL) based family analysis model. Testing on a 2018–2020 malware dataset classified by behavior indicated that combined characteristics achieved good accuracy. The malware detection approach by Cilleruelo *et al.* [21] used application static analysis and innovative training and dataset building. Google Play application lifespans were used to create a new dataset to prevent antivirus engine biases. The novel detecting mechanism differed from prior engines. Using 91,000 Google Play Store apps, experimental findings indicated 90% accuracy. Li *et al.* [22] suggested a factorization machine-based Android malware classifier that extracts features from manifest files and source code. Precision was high on the DREBIN dataset. Zhou *et al.* [23] suggested a static SIMGRU-based Android malware detection solution. It enhanced the gated recurrent unit (GRU) with similarities, creating new structures. These structures fared better than GRU models and other approaches in experiments. Alswaina and Elleithy [24] created RevEng, which provided application rights to ML algorithms. They used heavily randomized trees to decrease permissions, improving accuracy and execution speed. Two permission representation methods were tested: binary and weighted depending on feature relevance. Kumar *et al.* [25] described a malware detection system using ML, DL, and behavior and signature-based methodologies. It identified detection issues, classified ML methods, studied fundamental tactics, and examined DL. The RF model outperformed five other methods with good accuracy.

## 2. METHOD

Figure 1 depicts malware detection methodology used in this paper. This work contributed to the ongoing efforts to combat malware threats in the Android ecosystem by introducing a novel approach that leveraged image-based feature representations and ML algorithms. Through experimental evaluation and comparative analysis, the effectiveness of the method was demonstrated in accurately identifying malicious apps and mitigating the risks associated with mobile malware. The dataset used in this work consists of samples of both malware and benign apps collected from the virus share website, ensuring a diverse and representative set of app samples for analysis. After preprocessing the apps and extracting GIST features, a range of ML algorithms, including logistic regression (LR), KNN, AdaBoost, DT, NB, RF, SVM, extra tree classifier (ETC), and gradient boosting (GB), were applied to classify the apps based on their feature representations.

### 2.1. Android apps collection

Malicious mobile applications were gathered from virusshare.com. Malware apps are collected from two different datasets in virusshare namely vs2012 and vs2016. The number of malware samples in both datasets are 2,000 and 2,000 respectively. For benign applications; 1,700 samples were collected from CICAndMal2017 [26], while an additional 300 benign applications were sourced from the Google Play Store. With these apps, two datasets created each with 2,000 malware and 2,000 benign apps.

### 2.2. App to image conversion

Android apps are typically zipped files with .apk extension. Grayscale images were created in two ways: directly converting apk files and extracting dex files from apk files to generate grayscale images. The procedure involves opening a directory with APK/DEX files. For each file, the size is calculated. Depending on the size, the width of the image is determined (ranging from 32 to 1024). The file is then converted into an image using Python NumPy's "fromfile" method and saved as a .png file with the determined width using the "imsave" method. Finally, the directory is closed. This process applied in two different settings. In first case, the entrire Android apps are converted to images. In second case, only dex files of apps are converted to images.

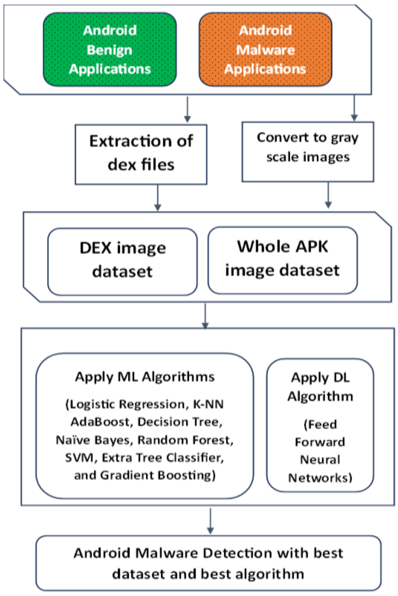


Figure 1. Proposed method

### 2.3. Extracting GIST features

The GIST descriptor, which is based on wavelet decomposition of an image, was utilized to compute texture features from the grayscale images. The extraction process was shown in Figure 2. This feature has been successful in scene classification and object classification. GIST global descriptors are useful for comparing images based on their content. After creating grayscale images for all malware and benign files, GIST features are extracted from images. The GIST features, typically employed in categorizing scenes such as forests, streets, and mountains, were adapted for use with Android APK files. The local representation of the image is then given by: vL (x)= vk (x)| k=1…N, here N is the number of sub-bands (N=20 taken). For the extraction of GIST descriptor texture features from the grayscale images, the Python "pyleargist" package was utilized. First 320 texture features for all the images are extracted and these feature vectors are converted to csv files.

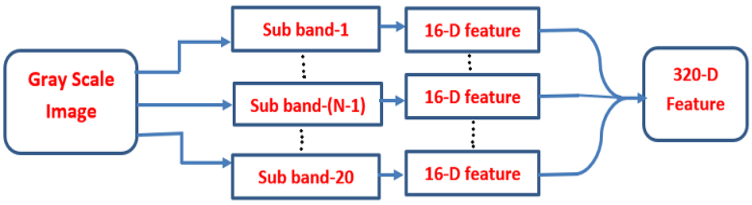


Figure 2. GIST descriptor feature extraction

Algorithm 1 applied to Android appls colleced including malware and benign apps. After applying Algorithm 1, two csv datasets created one with vs2012 malware samples and another with malware vs2016 samples. In both thr dataset, same benign samples are used.

Algorithm 1. Extraction of GIST features from gray scale images

```
Step 1: Open directory with gray-scale images
Step 2: For all the files in the directory:
Step 2.1: X = numpy. zeros((sum(no_of_imgs), 320))
  #Feature Matrix with 320 features
Step 2.2: i = 0
Step 2.3: Open all images with .png extension
Step 2.4: Apply python leargist package
Step 2.5: X[i]= des [0:320], i=i+1
Step 2.6: Convert X to data frame and then convert it in to csv file
Step 3: Close the directory
```

## 3. RESULTS AND DISCUSSION

After extracting GIST features from Android APKs, a range of linear ML classification algorithms was employed for malware detection. Subsequently, a DL feed-forward neural network algorithm was also utilized. Evaluation encompassed both ML and DL algorithms using varying numbers of samples of malware and benignware APKs. While linear ML algorithms achieved an accuracy of over 96%, the DL algorithm demonstrated consistent accuracy and recall rate across all experiments. These experiments were conducted using two different setups. In the first method, the GIST features of the entire APK images were utilized, whereas in the second case, only the image features of the DEX files were considered. In both cases feed forward neural network outperforms linear classification techniques.

### 3.1. Apply ML and DL algorithms with dataset1 (whole apk image to GIST feature dataset)

In this step, several ML classifiers applied with whole apk image to GIST dataset and the results are shown in Table 1. In Table 1, P indicates precision, R indicates recall and A indicates accuracy. From Table 1, it is observed that FFNN and RF gven 99%, 90% highest accuracy rate with good precision and recall rate for the two datasets. Next, GB, ETC, and Adaboost performed well with good accuracy.

Table 1. Results with ML algorithms (whole apk taken as grayscale image)

| Algorithm | Dataset1-A (malware: 2,000 from VS-2012, Benign:2,000) | | | Dataset1-B (malware:2,000 (from VS-2016, Benign: 2,000) | | |
|---|---|---|---|---|---|---|
| | P | R | A | P | R | A |
| LR | 97 | 91 | 95 | 91 | 60 | 80 |
| K-NN | 95 | 94 | 95 | 90 | 70 | 83 |
| AdaBoost | 98 | 95 | 97 | 84 | 68 | 80 |
| DT | 97 | 95 | 96 | 76 | 72 | 77 |
| NB | 99 | 86 | 93 | 83 | 67 | 81 |
| RF | 98 | 96 | 98 | 87 | 77 | 85 |
| SVM | 100 | 86 | 93 | 85 | 50 | 75 |
| ETC | 98 | 95 | 97 | 89 | 74 | 85 |
| GB | 99 | 95 | 97 | 76 | 77 | 80 |
| FFNN | 97 | 99 | 98 | 88 | 92 | 90 |

For dataset-1A and dataset-1B, a FFNN with the configuration in Figure 3 achieved better results. It has one input layer, 2 hidden layers, and one output layer. The number of neurons in each layer was 160, 81, 42, and 1 respectively. A batch size of 5 and 250 epochs were considered for training. Dropout, a method utilized to prevent overfitting, was implemented by adding one dropout layer to the model. This configuration resulted in a recall of 99% and an accuracy of 98% for dataset-1A and recall of 92% and accuracy of 90% for dataset-1B.

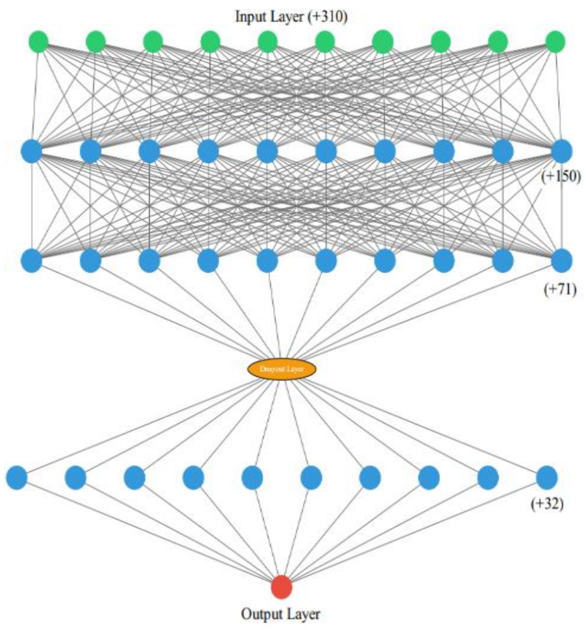


Figure 3. Feed forward neural network model (apk files as gray scale images)

### 3.2. Apply ML and DL algorithms with dataset2 (apk dex image to GIST feature dataset)

In this step, several ML classifiers applied with apk dex image to GIST dataset and the results are shown in Table 2. From Table 2, it is observed that FFNN given 98.7% and 98.8% highest accuracies with dataset2-A and dataset2-B. The precision and recall also good for FFNN. Next RF performed well with 98%, 98% for dataset2-A and dataset2-B.

For these datasets, a feed forward neural network with the configuration in Figure 4 achieved superior results. It has one input layer, three hidden layers, and one output layer. The number of neurons in each layer was 160, 80, 40, 21, and 1 respectively. A batch size of 6 and 250 epochs were considered for training. Additionally, one dropout layer was added to the model.

Table 2. Results with various ML algorithms (apk dex file taken as grayscale image)

| Algorithm | Dataset2-A (malware: 2,000 from VS-2012, Benign:2,000) | | | Dataset2-B (malware: 2,000 from VS-2012, Benign:2,000) | | |
|---|---|---|---|---|---|---|
| | P | R | A | P | R | A |
| LR | 97 | 94 | 95.9 | 97 | 93 | 95.9 |
| K-NN | 95 | 98 | 97 | 97 | 99 | 97.9 |
| AdaBoost | 96 | 99 | 97.7 | 98 | 98 | 97.5 |
| DT | 94 | 99 | 97 | 95 | 98 | 96.6 |
| NB | 97 | 93 | 95.5 | 98 | 94 | 96 |
| RF | 97 | 99 | 98 | 97 | 99 | 98 |
| SVM | 98 | 92 | 98.5 | 96 | 98 | 96.8 |
| ETC | 98 | 99 | 98.5 | 97 | 98 | 97.9 |
| GB | 98 | 93 | 95.9 | 97 | 98 | 97.6 |
| FFNN | 93 | 100 | 98.7 | 99 | 99 | 98.8 |

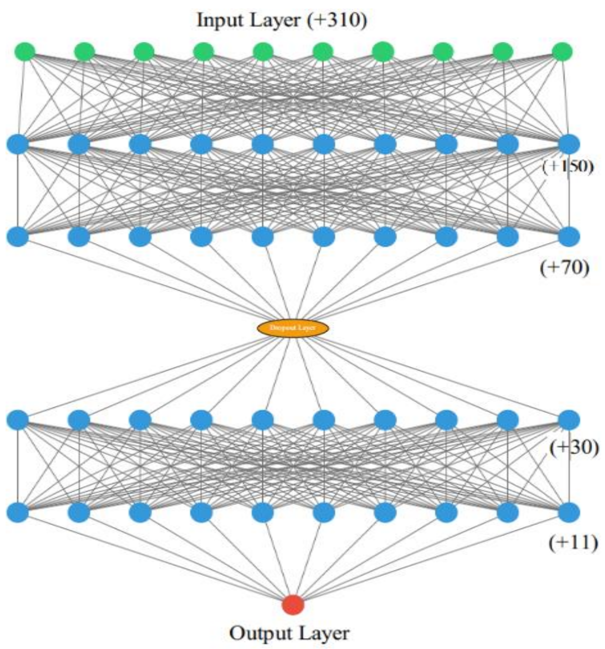


Figure 4. FFNN model (dex files as gray scale images)

### 3.3. Proposed methodology performance evaluation

Table 3 and Figure 5 shows performance comparison of proposed method with existing works. In comparison with existing methods, the proposed GIST-based approach demonstrates superior performance in malware detection accuracy. Conventional ML methods [1] achieve a respectable accuracy of 97%, providing a solid baseline for evaluation. SVM [3], another widely used technique, performs slightly lower with an accuracy of 96.6%. Odat and Yaseen [5], RF achieved a competitive accuracy of 98%. In the proposed method, the FFNN with whole image dataset given good accuracy of 98%. However, the proposed GIST-based approach for lex image dataset with FNN technique outperformed all existing methods, achieving an impressive accuracy of 98.8%. This substantial improvement signifies the efficacy of the GIST-based approach in enhancing accuracy in malware detection tasks.

Table 3. Comparison with existing work

| Model | Accuracy |
|---|---|
| Conventional ML [1] | 97% |
| SVM [3] | 96.6% |
| RF [5] | 98% |
| Proposed GIST based approach (lex images dataset) | 98% |
| Proposed GIST based approach (whole apk images dataset) | 98.8% |

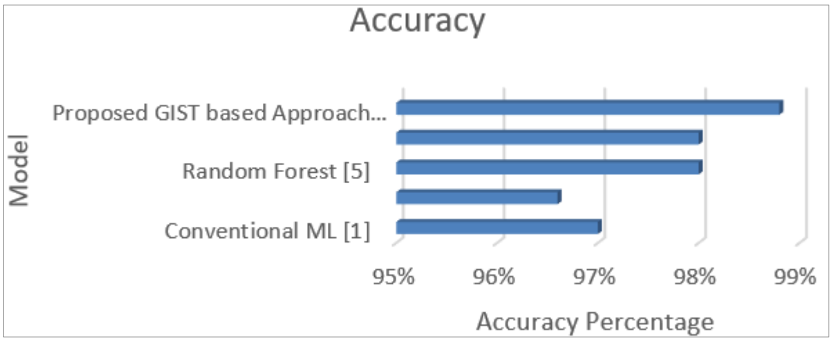


Figure 5. Performance evaluation of proposed model

## 4. CONCLUSION

In conclusion, this paper addresses the growing concern of malware presence in Android apps, given the significant role these devices play in both personal and professional spheres. With millions of apps available on the Play Store, detecting malicious software presents a formidable challenge. The proposed method leverages GIST features extracted from grayscale images of Android apps, providing a unique approach to malware detection. By employing various ML algorithms and a FFNN, superior accuracy was attained in classifying apps based on their features. The FFNN, in particular, demonstrated effectiveness in detecting malware compared to traditional classifiers. Overall, the proposed method offers a robust framework for enhancing Android security, empowering users to identify and mitigate potential threats effectively. Future work involves exploring scalability and efficiency to handle the growing volume of Android apps. Additionally, integrating real-time threat intelligence and behavioral analysis techniques aims to enhance malware detection accuracy and effectiveness on Android devices.

## REFERENCES

[1] S. Roy, S. Bhanja, and A. Das, "AndyWar: an intelligent Android malware detection using machine learning," *Innovations in Systems and Software Engineering. Springer Science and Business Media LLC*, Jul. 06, 2023. doi: 10.1007/s11334-023-00530-5.

[2] M. N.U.R. Chowdhury, A. Haque, H. Soliman, M. S. Hossen, T. Fatima, and I. Ahmed, "Android malware detection using machine learning: a review." *arXiv*, 2023. doi: 10.48550/ARXIV.2307.02412.

[3] M. Awais, M. A. Tariq, J. Iqbal and Y. Masood, "Anti-ant framework for Android malware detection and prevention using supervised learning," *2023 4th International Conference on Advancements in Computational Sciences (ICACS)*, Lahore, Pakistan, 2023, pp. 1-5, doi: 10.1109/ICACS55311.2023.10089629.

[4] F. A. Almarshad, M. Zakariah, G. A. Gashgari, E. A. Aldakheel and A. I. A. Alzahrani, "Detection of Android malware using machine learning and siamese shot learning technique for security," in *IEEE Access*, vol. 11, pp. 127697-127714, 2023, doi: 10.1109/ACCESS.2023.3331739.

[5] E. Odat and Q. M. Yaseen, "A novel machine learning approach for Android malware detection based on the co-existence of features," in *IEEE Access*, vol. 11, pp. 15471-15484, 2023, doi: 10.1109/ACCESS.2023.3244656.

[6] H. Alamro, W. Mtouaa, S. Aljameel, A. S. Salama, M. A. Hamza and A. Y. Othman, "Automated Android malware detection using optimal ensemble learning approach for cybersecurity," in *IEEE Access*, vol. 11, pp. 72509-72517, 2023, doi: 10.1109/ACCESS.2023.

[7] L. D. Costa and V. Moia, "A lightweight and multi-stage approach for Android malware detection using non-invasive machine learning techniques," in *IEEE Access*, vol. 11, pp. 73127-73144, 2023, doi: 10.1109/ACCESS.2023.3296606.

[8] A. Banik and J. P. Singh, "Android malware detection by correlated real permission couples using FP growth algorithm and neural networks," in *IEEE Access*, vol. 11, pp. 124996-125010, 2023, doi: 10.1109/ACCESS.2023.3323845.

[9] A. Mahindru, S. K. Sharma and M. Mittal, "YarowskyDroid: semi-supervised based Android malware detection using federation learning," *2023 International Conference on Advancement in Computation & Computer Technologies (InCACCT)*, Gharuan, India, 2023, pp. 380-385, doi: 10.1109/InCACCT57535.2023.10141735.

[10] A. Lakshmanarao and M. Shashi, "Android malware detection with deep learning using RNN from opcode sequences," *International Journal of Interactive Mobile Technologies (iJIM), vol. 16, no. 01. International Association of Online Engineering (IAOE)*, pp. 145–157, Jan. 18, 2022. doi: 10.3991/ijim.v16i01.26433.

[11] A. Subash, S. S. Rex, G. Vijay, G. S. R. E. Selvan, and M. P. Ramkumar, "Malware detection in Android application using static permission," in *Proceedings of the 5th International Conference on Inventive Research in Computing Applications, ICIRCA 2023*, Aug. 2023, pp. 1241–1245, doi: 10.1109/ICIRCA57980.2023.10220934.
[12] A. Sharma and H. Babbar, "An analysis of Android malware and IoT attack detection with machine learning," *2023 3rd International Conference on Intelligent Technologies (CONIT)*, Hubli, India, 2023, pp. 1-5, doi: 10.1109/CONIT59222.2023.10205931.
[13] A. S. Shatnawi, Q. Yassen, and A. Yateem, "An Android malware detection approach based on static feature analysis using machine learning algorithms," Procedia Computer Science, vol. 201. Elsevier BV, pp. 653–658, 2022. doi: 10.1016/j.procs.2022.03.086.
[14] A. Droos, A. Al-Mahadeen, T. Al-Harasis, R. Al-Attar and M. Ababneh, "Android malware detection using machine learning," *2022 13th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, 2022, pp. 36-41, doi: 10.1109/ICICS55353.2022.9811130.
[15] A. Bandi and L. Sherpa, "Android malware detection using machine learning classifiers," *Computer Networks and Inventive Communication Technologies. Springer Nature Singapore*, pp. 191–200, Oct. 14, 2022. doi: 10.1007/978-981-19-3035-5_15.
[16] H. Alkahtani and T. H. H. Aldhyani, "Artificial intelligence algorithms for malware detection in Android-operated mobile devices," *Sensors*, vol. 22, no. 6. MDPI AG, p. 2268, Mar. 15, 2022. doi: 10.3390/s22062268.
[17] A. Lakshmanarao and M. Shashi, "An efficient Android malware detection framework with stacking ensemble model," *International Journal of Engineering Trends and Technology*, vol. 70, no. 4. Seventh Sense Research Group Journals, pp. 294–302, Apr. 25, 2022. doi: 10.14445/22315381/ijett-v70i4p226.
[18] Y. Kanchhal and S. Murugaanandam, "Android malware a oversight on malware detection using machine learning," *2022 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, 2022, pp. 1-5, doi: 10.1109/ICCCI54379.2022.9741025.
[19] M. M. Alani and A. I. Awad, "PAIRED: an explainable lightweight Android malware detection system," in IEEE Access, vol. 10, pp. 73214-73228, 2022, doi: 10.1109/ACCESS.2022.3189645.
[20] Y. Ban, S. Lee, D. Song, H. Cho and J. H. Yi, "FAM: featuring Android malware for deep learning-based familial analysis," in IEEE Access, vol. 10, pp. 20008-20018, 2022, doi: 10.1109/ACCESS.2022.3151357.
[21] C. Cilleruelo, Enrique-Larriba, L. De-Marcos and J. -J. Martinez-Herráiz, "Malware detection inside app stores based on lifespan measurements," in *IEEE Access*, vol. 9, pp. 119967-119976, 2021, doi: 10.1109/ACCESS.2021.3107903.
[22] C. Li, K. Mills, D. Niu, R. Zhu, H. Zhang and H. Kinawi, "Android malware detection based on factorization machine," in *IEEE Access*, vol. 7, pp. 184008-184019, 2019, doi: 10.1109/ACCESS.2019.2958927.
[23] H. Zhou, X. Yang, H. Pan and W. Guo, "An Android malware detection approach based on SIMGRU," in *IEEE Access*, vol. 8, pp. 148404-148410, 2020, doi: 10.1109/ACCESS.2020.3007571.
[24] F. Alswaina and K. Elleithy, "Android malware permission-based multi-class classification using extremely randomized trees," in *IEEE Access*, vol. 6, pp. 76217-76227, 2018, doi: 10.1109/ACCESS.2018.2883975.
[25] A. Kumar, K. Abhishek, S. K. Shandilya and M. R. Ghalib, "Malware analysis through random forest approach," in *Journal of Web Engineering*, vol. 19, no. 5-6, pp. 795-818, September 2020, doi: 10.13052/jwe1540-9589.195610.
[26] UNB Cybersecurity Research Group, "UNB-CIC: Android malware dataset," [Online]. Available: https://www.unb.ca/cic/datasets/andmal2017.html. Accessed on January 27th, 2024.

## BIOGRAPHIES OF AUTHORS

**Dr Ponnuswamy Udayakumar** received his M.Tech. degree with Hons and a Gold Medal in CSE from Chhattisgarh Swamy Technical University, Bhilai (C.G.). Earned his Ph.D. in CSE from MATS University, Raipur (C.G.). Currently serving as a professor in the department of CSE at Akshya College of Engineering and Technology, Coimbatore. He has published more than 40 research papers in various national, international journals including those indexed in Scopus, WoS and UGC journals. He has published three text books and holds seven patents both filed and published. He has 22 years of teaching experience, and holds lifetime memberships in ISTE, CSI and ASDF. He has received Best Professor Award Globally in the year 2022 by ASDF. His research interest includes wireless sensor networks, network security, artificial intelligence, and data science. He can be contacted at email: drudaycse@gmail.com.

**Srilatha Yalamati** is Assistant Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh. She is pursuing Ph.D. in Gandi Institute of Technology and Management, Visakhapatnam. She has 14 Years of teaching experience and 3 years of industry experience. She published papers in reputed national and international journals. Her areas of interest include datamining, machine learning, computer networks, and artificial intelligence. She can be contacted at email: srilatha.yalamati@gmail.com.

**Dr. Lavadiya Mohan** is Associate Professor, Department of CSE (IoT) in Balaji Institute of Technology Science, Telangana India. He received his Ph.D. from Rayalaseema University in Datamining. He has more than 15 years of teaching experience. He had published papers in reputed national and international journals. He had attended many workshops, conferences and presented various research papers at national and international conferences. His areas of interest include datamining, software engineering, computer networks, and artificial intelligence. He can be contacted at email: lavmohan@gmail.com.

**Dr. Mohd Junedul Haque** currently working as Associate Professor in School of Computer Sciences and Engineering Sandip University Nashik Maharashtra. He completed his M. Tech in Information Technology from GGSIP University Delhi and Ph.D. in CSE from OPJS University Rajasthan. His areas of interest are artificial intelligence and data base management system. He can be contacted at email: mohammad.haque@sandipuniversity.edu.in.

**Gaurav Narkhede** currently working as Assistant Professor Department of Electrical and Electronics Engineering, Dr Vishwanath Karad MIT World Peace University Pune. He completed his B.E in E and TC and M. Tech in Communication Engineering from NITK, Surathkal. His areas of interest are machine learning and deep learning. He can be contacted at email: ggnarkhede9@gmail.com.

**Dr. Krishna Mohan Bhashyam** is having 20+ years teaching experience and presently working as an Associate Professor, Department of Information technology in R.V.R and J. C. College of Engineering, Guntur, Andhra Pradesh. He obtained his Ph.D. from Bharathidasan University, Trichy in the stream of Computer Science and Engineering. His research works on time series forecasting, using deep learning techniques. He published 2 SCIE indexed journals,5 Scopus journals,1 ABDC indexed journals and 10 more research articles published in various international journals. He authored 2 text books and also published 3 patents. He can be contacted at email: mohanbk28@gmail.com.