ANDROID MALWARE DETECTION USING GRAPHICAL TECHNIQUES

A MAJOR PROJECT-II REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
INFORMATION SYSTEMS

Submitted by:
MONISH KUMAR SAHU
2K22/ISY/09

Under the supervision of

MR. RAHUL GUPTA

DEPARTMENT OF INFORMATION AND TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

May, 2024

ACKNOWLEDGEMENT

I would like to express my sincere gratitude towards my supervisor Mr. Rahul Gupta for providing her invaluable guidance, comments, and suggestions throughout the course of the project.

The results of this thesis would not have been possible without support from all who directly or indirectly, have lent their hand throughout the course of the project. I would like to thank my parents and faculties of the Department of Information Technology, Delhi Technological University, for their kind cooperation and encouragement which helped me complete this thesis. I hope that this project will serve its purpose to the fullest extent possible.

MONISH KUMAR SAHU
2K22/ISY/09

DELHI TECHNOLOGICAL UNIVERSITY
(FORMERLY DELHI COLLEGE OF ENGINEERING)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Monish Kumar Sahu, 2K22/ISY/09 student of M.Tech in Information Systems, hereby declare that the Major Project-II dissertation titled "**ANDROID MALWARE DETECTION USING GRAPHICAL TECHNIQUES**" which is submitted by me to the Department of Information Technology, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any degree, Diploma Associateship, Fellowship, or other similar title or recognition.

Place: Delhi **MONISH KUMAR SAHU**
Date: 20/05/2024 **2K22/ISY/09**

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(FORMERLY DELHI COLLEGE OF ENGINEERING)
Bawana Road, Delhi-10042

CERTIFICATE

I hereby certify that the Major Project-II dissertation titled "**ANDROID MALWARE DETECTION USING GRAPHICAL TECHNIQUES**" which is submitted by Monish Kumar Sahu, 2K22/ISY/09, Department of Information Technology, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any degree or diploma to this University or elsewhere.

Place: Delhi
Date: 20/05/2024

MR. RAHUL GUPTA
SUPERVISOR
Asst. Professor
Department of Information Technology
DELHI TECHNOLOGICAL UNIVERSITY

ABSTRACT

Android is the most popular operating system for mobile devices which has dominated the smartphone industry. Traditional signature-based malware detection approaches have been in use for a long time. Yet their technology falls far short of being completely secure. Modern malware detection tools are used by major mobile application distributors, official stores, and marketplaces to analyze uploaded programs and eliminate any dangerous ones. Unfortunately, until they are taken off the market, malicious software has a long window of opportunity. In this paper, we studied different research papers based on graphical techniques and learned about static and dynamic malware detection approaches. We presented a new and unique method for detection of Android malware that uses apk function analysis and app-similarity graph in conjunction with ensemble techniques. This work uses graph neural networks and similarity graph, in which relationships are depicted as edges based on semantic similarities, while apps are represented as nodes. The study shows how well the suggested technique works to recognize malicious apps by comparing their functional and structural characteristics. The results improve the field of mobile app security and present an effective strategy of action to protect against threats that are constantly evolving within the Android app ecosystem. The proposed model provided reasonable accuracy and hence served to aid and maintain a user-safe environment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 GENERAL

Due to their portability, high processing power, accessibility to the Internet, and usability, smartphones, tablets, and other mobile platforms have been an integral part of our everyday lives for a long time and are now considered essential tools in our society. Currently, sales of personal computing devices like tablets or smartphones reached to 1433 million units in 2021 [1], which leads to the encouragement to develop more and advanced mobile malwares. New online dangers appear every year. The Atlas VPN team revealed data showing that over 34.76 million new malware samples have been found so far this year. This indicates that in 2022, hackers have produced more than 316 thousand new malware threats every day on average [2].

According to data provided by AV-TEST GmbH, an independent provider of services in the areas of IT Security and Anti-virus Research, the analysis was conducted. The information was last updated on April 20, 2022.

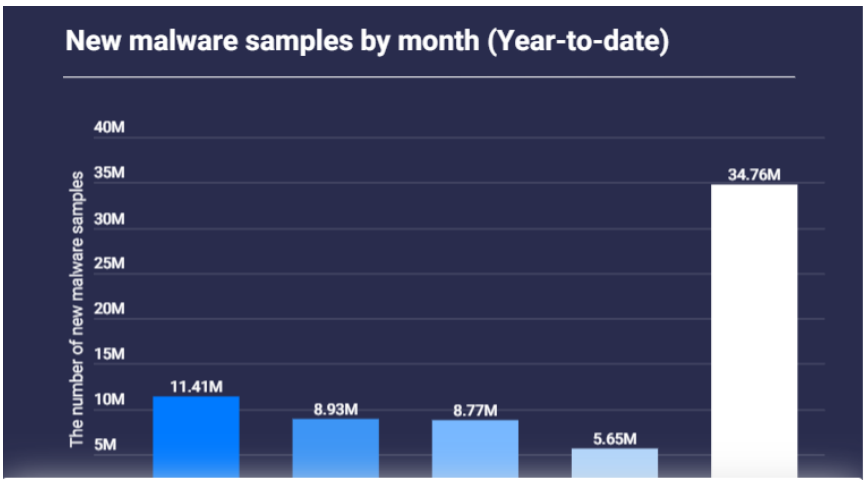


Fig 1.1 Number of new malware samples in 2022.

1.2 ANDROID MARKET SURVEY

With Android being the most used mobile operating system (OS), with roughly 71% of the global market share as of November 2022 [3].

Fig. 1.2 Mobile OS Market Share Worldwide

Android has an open-source philosophy and offers developers a free integrated programming environment (IDE) that makes it easier for them to use its platform. On the other side, iOS has a higher entrance hurdle for those looking to build for the iOS ecosystem because of its strict approval rules and need that developers utilise proprietary hardware and software to create and publish iOS apps.

The Android operating system also enables users to download programmes from unreliable sources that could be found online and via third-party app stores.

Latest survey says, in 2022 there were more than 5 million apps available in top application markets (Google Play, iOS AppStore, and Amazon Appstore) [4][5].

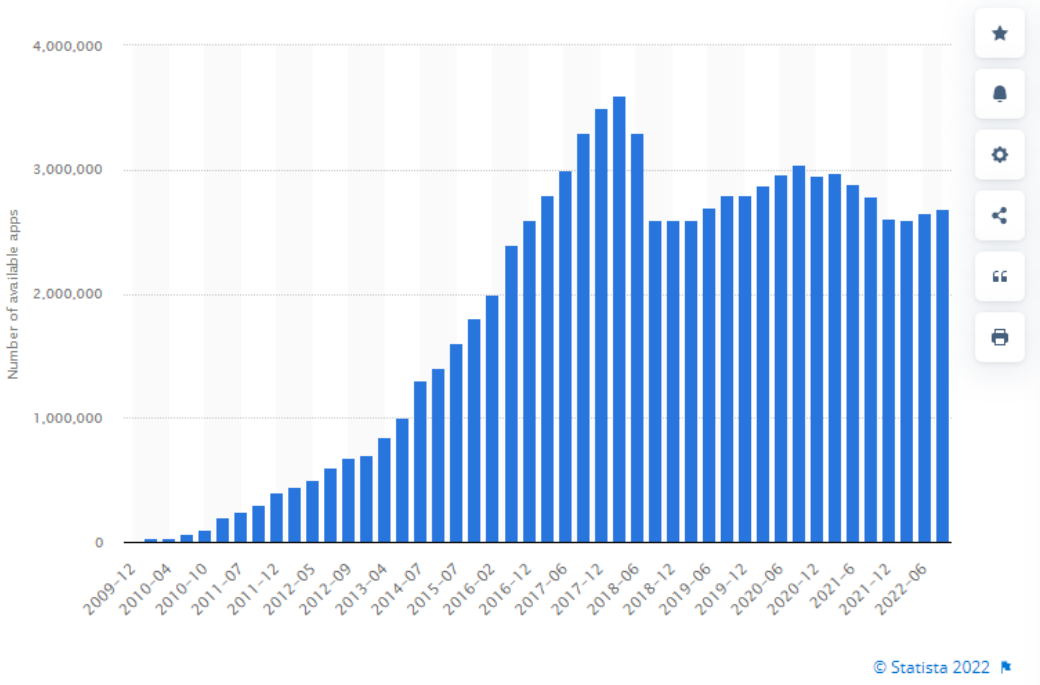


Fig. 1.3 Number of available applications in the Google Play Store

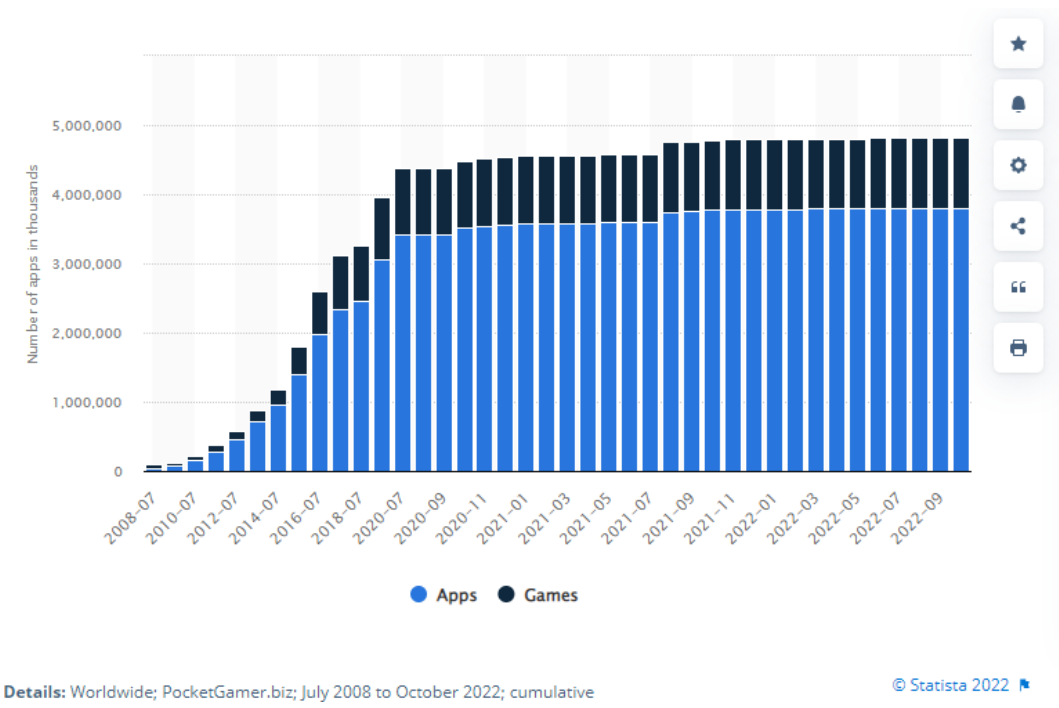


Fig. 1.4 Number of available apps in the Apple App Store from 2008 to October 2022.

1.3 ANDROID MALWARE DETECTION APPROACHES

Mobile devices, particularly smartphones and tablets, have become a crucial part of modern life, especially in developed countries. Their widespread use and sophisticated features make them attractive targets for attackers. As a result, protecting private user information and sensitive data requires strong security measures in the digital age. Android applications are classified as harmful or benign, with harmful ones posing security risks. The increasing use of Android by cybercriminals has led to the development of harmful apps that steal data. Users can install apps from untrusted sources, increasing the risk of malware infection. Recent studies show that 97% of mobile malware infections occur on Android devices. Malware can lower system performance, scan OS vulnerabilities, and perform undesired operations, making malware analysis a critical cyber-security issue.The Android platform is susceptible to malicious applications, leading to various studies to develop methods for detecting and removing them. These methods include static and dynamic analysis, as well as a hybrid analysis approach that combines both methods.

Android platform is vulnerable to malicious applications. Various Studies have been carried out to develop the method to detect such malware in android platforms. Performing the analysis after determining which features can be taken out of the apps. Static and dynamic analysis are the two primary categories into which these analytic approaches fall. There is a hybrid analysis approach also, which is a mixture of both other approaches.

1. Static analysis

Static analysis is a method of learning about software without running the application, utilizing information such as opcode sequences and control flow graphs obtained from disassembling binary files [1]. These feature sets can be combined or used singly to identify malware [2]. Extensive static testing of manifest about the app and internal commands is done by the Drebin [3], DroidMat [4], DroidAPIMiner [5], DroidSieve [6], and DroidDet [7] methods in

order to gather sensitive permissions, API calls, and other distinguishing aspects. Later, Drebin [3] also produces vectors for each app using the features that have been gathered.

2. Dynamic analysis

We must run the software, frequently in a virtual environment, in order to do dynamic analysis. Substantial information on a variety of topics, including memory writes, registry modifications, API calls, system calls, and instruction traces is provided by dynamic analysis [1]. In a fake environment that mimics a real device, Andlantis [8] does dynamic analysis. To find anomalous activity, forensic imprints of malware families, runtime behaviors, and system calls are gathered. [1].

3. Hybrid methods

The pros of both static testing and dynamic testing are combined in hybrid approaches. We talk about two recent instances of this kind of work in this section. The authors suggest a system for categorising malware using static and dynamic analysis. They use a method they call Malware DNA to characterise the characteristics of malware (Mal-DNA). This method's core component is a behaviour monitor and analyzer based on debugging that pulls out dynamic features. A hybrid system called IntelliDroid [9] combines static and dynamic analysis and creates inputs tailored to a dynamic analysis tool. The process of generating input improves code coverage, which raises the likelihood of identifying whether the functionality that was performed was benign or malicious [10].

Hybrid analysis combines static and dynamic analysis, removing compromise signs and refining malware properties. This approach offers a comprehensive understanding of malware but has higher resource needs and the need to adapt to malware's ongoing development. Combining these techniques allows security

experts to build multi-layered protection and conduct in-depth analysis of potential threats.

Based on the facts provided above, it is evident that an efficient way to identify Android malware is desperately needed in order to shield Android users from harmful assaults.

This work makes six distinct contributions:

- One of the tasks is creating a relationship graph that shows the relationships between various apps and their functionalities.
- This study makes use of effective methods to determine how similar different applications are to one another, which helps to provide a more thorough knowledge of their features and attributes.
- We offer a scalable technique for assessing app-function correlations using mature and well efficient recommender system modules.
- The suggested method does not need expert-based feature extraction. Automatic feature generation involves semantic node embedding of the ASG.
- Our work is experimented on CIC-AndMal2017 dataset which is an extensive compilation of both malicious and benign android apps.
- This study makes use of the CIC-AndMal2017 dataset. This dataset provides the experimental basis for evaluating the suitability and efficacy of the suggested approaches.

The remaining sections of the paper are separated as, the second section gives introductory information and a review of the literature; the third section explains the suggested methodology for app categorization; the fourth section discusses datasets, implementation, and results; and the fifth section concludes the whole paper.

CHAPTER 2
RELATED WORK

This research [3] proposes a machine learning-based Android malware detection system to enhance smartphone users' security and privacy. This system categorizes Android programs as malware or goodware, using permission-based features and events. The framework uses a dataset, extracts attributes from Android.apk files, and empirically tests machine learning algorithms for high accuracy rates.

This research [4] introduces DL-Droid, a deep learning system that uses stateful input generation and dynamic analysis to detect harmful Android applications. Experiments on genuine devices with over 30,000 malicious apps showed DL-Droid outperforms traditional machine learning methods with recognition rates of up to 97.8% for dynamic characteristics and 99.6% for combining dynamic and static data. The study emphasizes the need for enhanced input generation for dynamic analysis in Android malware detection systems.

This study [5] presents a classification model for Android malware apps that uses permission requests and API calls. The model categorizes apps into disruptive, dangerous, and ambiguous, based on their use of API calls and risky permissions. The model, built on 27,891 Android applications from a malware dataset, has an F-measure of 94.3%, making it useful for malware analysis and forensic investigations. The method uses statistical tests to automatically assess and evaluate applications, revealing that permissions and API calls significantly impact malware variant classification. The technique also provides valuable insights into virus activity.

This study [6] introduces a new malware detection method that separates a function call graph into community structures for malware detection. It uses machine learning categorization instead of subgraph similarity comparison and reduces computing time. The method outperformed three well-known anti-virus software and two earlier control flow graph-based methods in various malware

families. It uses reverse engineering to retrieve an Android application's function call graph, weights it based on permissions and APIs, and detects malware using machine learning classification.

This study [7] for malware vetting tools presents a deep learning-based hybrid analysis strategy that combines the benefits of static and dynamic analysis to improve accuracy. The approach generates a number of artifacts using lightweight static and dynamic analysis procedures. These are then trained to create separate models, which are merged to form a hybrid classifier. Using hybrid analysis, the best deep learning model has an AUC (area under the precision-recall curve) of 0.9998. The study also examines the performance indicators of several deep learning framework modifications, demonstrating that the system is scalable and adaptable to imbalanced data sets. The paper also includes a comparison of the performance indicators for several deep learning framework versions.

This study [8] tells that a 2011 study found 211 harmful Android applications on the official Android market. Detecting malware using machine learning-based classifiers faces challenges in collecting feature representations and selecting a classifier that can only be trained in one category. To overcome these, a diverse feature set is retrieved and processed separately by kernels. A One-Class Support Vector Machine is trained on benign applications, using a server's computational capability.

This study [9] explores how mobile malware bypasses detection by replicating security-sensitive activities of innocuous programs and reducing their payload. The authors introduce AppContext, a static program analysis tool that helps distinguish between benign and malicious behavior by collecting contexts for security-sensitive actions. Tested on 633 benign applications and 202 malicious apps, the results show that the purpose of a security-sensitive action significantly impacts its maliciousness. The key contributions include an abstraction to describe security-sensitive activity contexts, a static-analysis approach for context extraction, and three assessments of 846 Android apps.

This work [10] introduces a new dynamic analysis approach called Component Traversal, which uses weighted directed graphs and a deep learning architecture to detect unknown An-droid malware. This method, which has been integrated into a commercial Android anti-malware application, outperforms other alternative strategies, as attackers are increasingly using tactics like repackaging and obfuscation to avoid signatures.

This study [11] presents a framework for dividing Android APKs into benign or harmful families using a model-based semi-supervised (MBSS) classification approach. MBSS outperforms traditional malware detection classifiers like SVM, kNN, and LDA under ideal classification settings. Its accuracy is 98% in-sample and has minimal false positive rate. Out-of-sample testing shows MBSS and SVM maintain a 90% detection rate, while kNN and LDA perform significantly worse. MBSS continues to outperform other classifiers.

This research [12] proposes a deep learning method called CDGDroid, which uses semantic graph representations like control flow graph and data flow graph. Android malware poses a threat to digital lives, necessitating effective detection and security. Recent machine learning algorithms often lack the complexity needed for Android applications. The model uses a convolutional neural network to build a classification model. Tests on various datasets show optimal accuracy when combined horizontally. CDG-Droid outperforms other anti-virus tools.

The growing amount of malware programs on Android, the leading smartphone operating system, poses a serious danger to consumer privacy and security. In this study [11] we studied about Classification algorithms using a single feature frequently have poor detection performance. To improve detection, the FAMD framework (Fast Android Malware Detector) is suggested in this work. The framework reduces feature dimensionality by using permissions and Dalvik opcode sequences from samples that have been preprocessed using the N-Gram approach and the FCBF algorithm. The dimensionality-reduced features are then fed into the CatBoost classifier for malware detection and family categorization. The findings demonstrate that the combined characteristics increase malware

detection accuracy by 97.4% on the Drebin dataset [3] and 97.4% on malware family classification accuracy. This framework surpasses other cutting-edge approaches in terms of accuracy and time consumption.

In this paper [12] researchers have developed new approaches for detecting Android malware based on syntactic traits and machine learning techniques. This study describes a novel technique to Android malware classification based on deep learning and OpCode-level Function Call Graph (FCG) [13]. The method employs a Long Short-Term Memory (LSTM) deep learning model and was evaluated on a dataset of 1796 Android malware samples and 1000 benign Android applications [13]. The findings revealed that the suggested methodology beats state-of-the-art approaches with 97% and 91% accuracy, respectively, while taking up less time.

In this study [14] four different malware detection techniques that use the Hamming distance to measure sample similarity are presented. These approaches include the K-Medoid Based Closest Neighbors (KMNN), Weighted All Nearest Neighbors (WANN), All Nearest Neighbors (ANN), and First Nearest Neighbors (FNN). Their objective is to alert users about an Android app's possible risks, reducing the possibility of malware spreading widely. The algorithms are evaluated on three datasets, which include both benign and malicious Android applications such as Drebin [3], Contagio, and Genome. Performance comparisons with cutting-edge algorithms such as Mixed and Separated solutions, PDME, and FalDroid demonstrate that the suggested methods achieve equivalent accuracy rates to existing solutions.

The study's [15] goal is to develop malware detection algorithms for Android applications due to their widespread use and harmful variations. The authors employ API call graphs to precisely represent application activity, however similarity detection and classification techniques can be sluggish and imprecise. They incorporate API call graphs in a deep neural network, which detects similarities between binary functions. By evaluating different embedding methodologies and altering network setup variables, the research aims to

optimize network performance. The results of the experiment show that the malware classification achieved 98.8% precision, 98.4% recall, 98.6% F-measure, and accuracy.

The paper [16] proposes, creating behavior characteristics of Android applications using complete, multi-view information using the apk2vec semisupervised Representation Learning (RL) framework. This method can considerably enhance downstream analytics activities such as app classification, recommendation, and virus detection. For effective online learning, the system includes information from numerous semantic perspectives, employs app-specific labels, and combines RL with feature hashing. The generated semi-supervised multi-view hash embeddings can be utilized for a variety of subsequent tasks. Experiments with almost 42k applications demonstrate that apk2vec's app profiles beat cutting-edge approaches in four app analytics tasks: virus identification, family grouping, app clone detection, and app recommendation.

This study [17] This paper suggests DLGraph, a novel graph embedding and deep learning-based malware detection method. The system learns computer program function-call graphs and Windows API calls using two stacked denoising autoencoders. It then merges latent representations to provide a feature vector for malware detection. Experiments on diverse datasets show that the strategy is effective and superior to a comparable method.

This study [18] presents a framework for detecting fraudulent applications for Android that is built on the concepts of Active Learning technologies and SVM (Support Vector Machine). The method captures application execution activity and translates them into a feature set, assigning timestamps to select features. The use of time-dependent activity tracking increases malware detection accuracy. The model was created using the Expected error reduction query approach and adaptive online learning. Experiments on the DREBIN benchmark [3] malware dataset demonstrate that the technique accurately detects harmful apps and improves updateability against new infections.

In this paper [6] we studied that current machine learning classification algorithms rely on lightweight syntactic cues; however, they may be ineffective for identifying obfuscated Android malware. In the paper DroidSieve is presented which is an Android malware classification system. It employs static testing to detect malicious apps and group them into virus families based on similarities. DroidSieve employs obfuscation-invariant characteristics and artifacts, achieving up to 99.8% accuracy for malware detection and 99.2% accuracy for obfuscated malware family detection.

This research [19] describes a machine learning approach based on n-opcode analysis for classifying and grouping Android malware. This method eliminates the requirement for expert knowledge to describe required features, and it has been tested on 2520 samples with up to 10-gram opcode features. The technique produced an f-measure of 98%, underlining the rising challenge of malware detection on the Android mobile platform, which is popular and accessible through third-party app stores.

The paper [7] provides a low-cost, high-efficiency strategy for detecting Android malware that involves extracting permissions, sensitive APIs, monitoring system events, and permission rates. The Ensemble Rotation Forest (RF) model is used to identify fraudulent Android applications. The approach yields 88.26% accuracy, 88.40% sensitivity, and 88.16% precision. The suggested strategy outperforms the state-of-the-art Support Vector Machine (SVM) model by 3.33%. The findings indicate that the suggested approach is extremely promising and might provide a cost-effective option for Android virus detection.

.

CHAPTER 3
METHODOLOGY

Our suggested Android app analysis and categorization approach consists of eight basic components.

1. **Function extraction**. The Android application archive (.apk/APK) contains the app's whole bytecode [1]. The bytecode is first converted into the source code of the application, after which all specified and utilized functions are extracted and saved.
2. **ASG formation**. The retrieved functions are processed to generate an app similarity graph (ASG). The ASG uses recommender system-based item similarity approaches to connect apps with comparable functionalities.
3. **SVM:** In high-dimensional spaces, SVMs function well. SVMs only need a subset of training data points known as support vectors to identify the ideal hyperplane also they are memory efficient.
4. **Gradient Boost:** XGBoost is an effective and versatile tool that can handle complex correlations and patterns in data because of its efficiency and adaptability.
5. **AdaBoost:** AdaBoost enhances model performance by prioritizing relevant features, making it beneficial for high-dimensional datasets and capturing complex relationships, making it suitable for weak learners like decision trees.
6. **Ensemble technique by MaxVoting -** Applying the max voting approach to combine the forecasts of the three models. This means deciding the majority vote from the forecasts of the individual models for each data point.
7. **Ensemble technique by Stacking -** Stacking is the process of training a meta-model that combines base model predictions. In this work, we are using Random Forest, SVM, and AdaBoost as basic models before training a meta-model to create the final predictions.

8. **Ensemble technique by Bagging -** Bagging (also known as Bootstrap Aggregating) is the process of training numerous instances of the same basic model on different subsets of training data and then merging their predictions.

3.1 Function extraction

Androguard's feature extraction includes examining Android application packages (APKs) to extract various characteristics or features. These features may be related to the app's structure, behavior, or functionality. Androguard is a well-known Python library that analyzes Android apps and provides utilities for working with APK files.

Here's an overview of the feature extraction process with Androguard:

- **Loading APK:** Load the APK file using Androguard's APK class. This class allows access to APK-specific information such as the manifest, resources, and classes.
- **Loading Dalvik Bytecode:** After loading the APK, locate the Dalvik bytecode in the DEX (Dalvik Executable) files. Androguard provides the DalvikVMFormat class to interact with DEX files.
- **Class and Method Extraction:** Extract app structure information by iterating through Dalvik bytecode classes and functions.
- **Feature Representation:** The obtained features may vary based on the analysis's specific needs. Depending on the analysis objectives, the features may include permissions, API calls, intent filters, and other APK components.
- **Error Handling:** During feature extraction, errors may occur. It is vital to handle exceptions gracefully. This prevents the script from crashing and allows you to identify and fix specific issues.

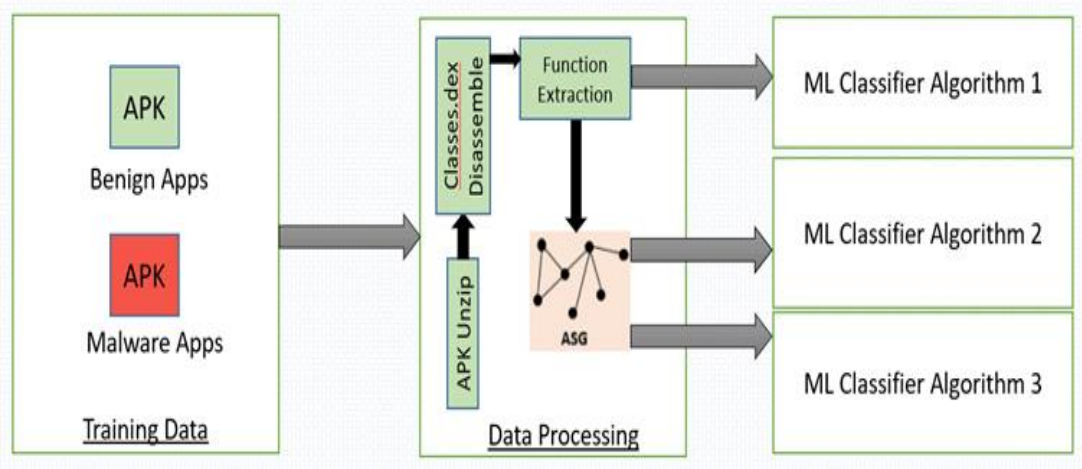


Fig. 3.1. Overall app analysis and classification.

3.2 App-Similarity Graph formation

We created the App Similarity Graph (ASG) to better app representation. To begin, we generate a bipartite graph using the popular methods from the previous step. The bipartite graph is indicated by GAF = (A, F, E), where A represents apps, F represents the important functions found in the previous step, and E represents the edges connecting each function to its own app. The GAF apps-functions network is evaluated using an ISR model to determine item similarity.

In recommender systems, an ISR model assesses the similarity of two items based on user actions. Pearson correlation, Jaccard similarity, and cosine similarity are three commonly used methods for measuring item similarity. In this paper, we employ the Cosine similarity approach.

A metric for comparing two non-zero vectors defined in an inner product space is called cosine similarity. The cosine of the angle between the vectors, or the dot product of the vectors divided by the product of their lengths, is what is known as cosine similarity. As a result, the cosine similarity solely depends on the angle of the vectors rather than their magnitudes. The equation for cosine similarity is given as:

$$\cos(\Theta) = A \cdot B / \| A \| \| B \|$$

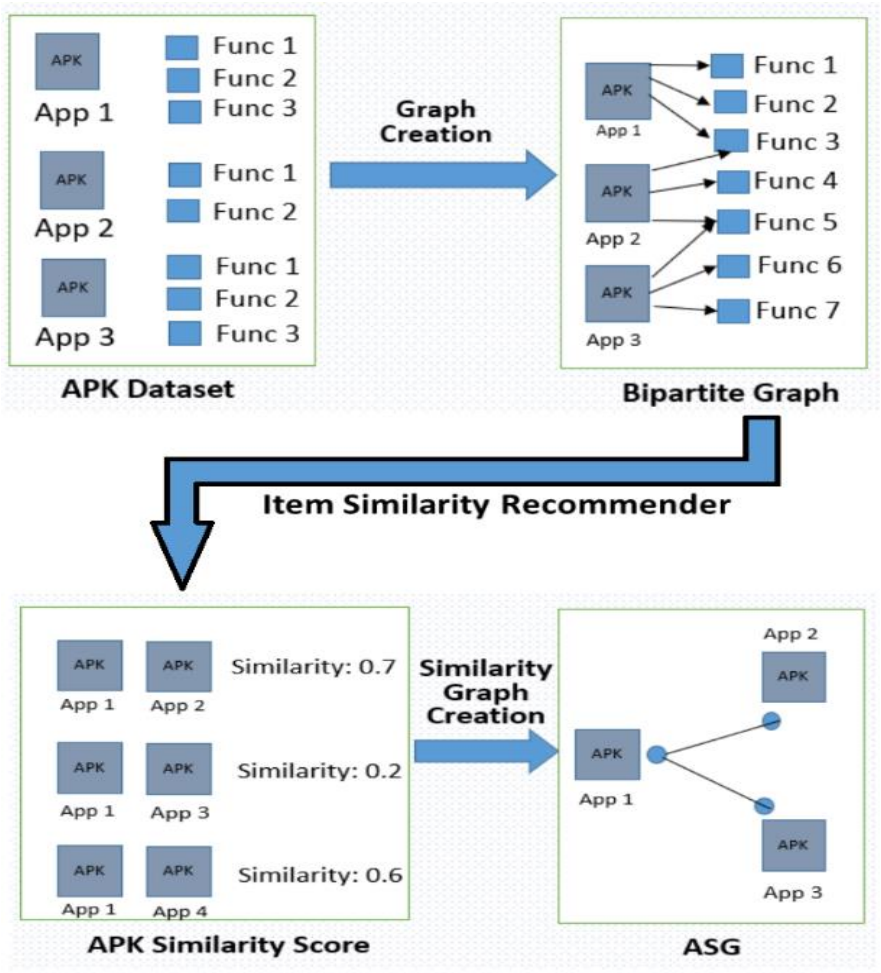


Fig. 3.2. App similarity graph creation process

3.3 ASGnode2vec + SVM classification

After ASG is created, we use a two-step classification process that consists of applying an SVM classifier to the automatically generated feature vectors and utilizing a node2vec to construct features on the ASG (ASGnode2vec).

An approach for feature learning in networks based on neural networks is called Node2vec [20]. Node2vec, which draws inspiration from word2vec [21], employs the wordcontext concept to provide a low-dimensional features representation for nodes. The acquired characteristics often maintain node network neighborhoods, which arranges the nodes in accordance with their functions within the network and the communities in which they are a part of.

Every app in the ASG has a continuous vector representation as the result of node2vec applied to the ASG. ASGnode2vec retrieves the semantic context for programs from the App Similarity Graph architecture by using the node2vec options [1]. The feature vectors produced by ASGnode2vec, which include both less and more comparable applications, are influenced by the ASG structure when intermediate $q \approx 1$.

After that, we build a relational dataset in which applications are instances and A SGnode2vec automatically generates feature vectors.
On the acquired vectors, a supervised machine learning algorithm is performed.

3.4 ASGnode2vec + Gredient boost classifier

A machine learning model called XGBoost creates embeddings for every app in a dataset using Node2Vec characteristics. Each app's connections with other applications in the graph are captured by these embeddings, which show each app as a dense vector. The process of feature extraction yields a feature matrix that is used as the XGBoost classifier's input. Each program is given a label according to its kind, resulting in a labeled dataset. The XGBoost model is trained on the training set of the dataset, while the testing set is used to assess the model's performance.

XGBoost successively constructs an ensemble of decision trees while training the model with the training data. To determine which Node2Vec properties are most important for the model's prediction performance during training, XGBoost computes feature importance. Apps may be categorized as benign or malicious using the trained XGBoost model, which generates probabilities for each class and makes predictions on the testing set.

Metrics including accuracy, precision, recall, F1-score, and ROC-AUC are used to assess performance. In order to differentiate between malicious and benign applications, the significance of the model is examined, offering insights into the specific elements of the Node2Vec embeddings that are critical for classification. Hyperparameters or settings are adjusted as needed to maximize the performance of the model.

3.5 ASGnode2vec + AdaBoost classification

Adaboost is an ensemble learning technique that builds a powerful classifier by combining several weak classifiers. It represents each app with a dense vector of characteristics by embedding nodes into a high-dimensional vector space using Node2Vec embeddings. The app's relationships with other nodes in the network are captured by these characteristics, which stand in for the structural and contextual information that Node2Vec learnt. These characteristics are used as input by Adaboost during training, and the related labels (benevolent or malevolent) are used as output.

Adaboost begins with a subpar classifier, frequently a straightforward decision tree, which might not function effectively by itself. By giving misclassified instances a larger weight, it enables weaker classifiers to concentrate on cases that are challenging to classify. By training each weak classifier on a portion of the data, Adaboost builds an ensemble of them. By giving each poor classifier a weight determined by its accuracy, each one adds to the final result. Weak classifiers that do well on challenging cases are given preference in the final classification, which is a weighted mixture of their individual outputs. Adaboost determines the final classification using a weighted majority vote.

3.6 Ensemble technique by MaxVoting

The integration of node2vec with ensemble learning via a max voting approach enhances app similarity predictions by combining the advantages of multiple machine learning models. The project begins with the graph em-bedding technique node2vec, which extracts features from Android APK files. The bipartite graph is formed by app interactions and node embeddings, efficiently capturing context and structural details. An ensemble learning strategy is then used, using three base models: Random Forest, Support Vector Machine (SVM), and Ada-Boost. The labels and node embeddings are trained using the cosine similarity matrix. The ensemble is then aggregated using the max voting approach.

The max voting ensemble gathers individual predictions from each base model, with the class that occurs most frequently for each data point selected as the final prediction. This technique enhances app similarity predictions by utilizing diverse viewpoints and capabilities. The ensemble uses Random Forest, SVM, and Ada-Boost models to capture intricate interactions in the bipartite graph. This integrated approach uses graph-based embeddings from node2vec for nuanced representation of app-function relationships and ensemble learning to enhance predictive accuracy. Max voting ensures a robust decision-making process, making final predictions more reliable and robust in capturing patterns of similarity between different Android apps. This comprehensive workflow showcases a holistic approach to app similarity analysis.

3.7 Ensemble technique by Stacking

After the node2vec embedding is generated, ensemble learning methods—specifically, stacking and bagging—are utilized to improve the model's predictive power. For this ensemble, three unique base models are selected: Random Forest, Support Vector Machine (SVM), and AdaBoost. Using the labels produced from the cosine similarity matrix, which indicate the similarity scores between various applications, each model is trained on the node embeddings. In stacking, the predictions of the three underlying models are combined by training a meta-model—in this example, another Random Forest. The ensemble may assess and learn from the advantages and disadvantages of each model through the stacking process, resulting in a more sophisticated and knowledgeable decision-making process.

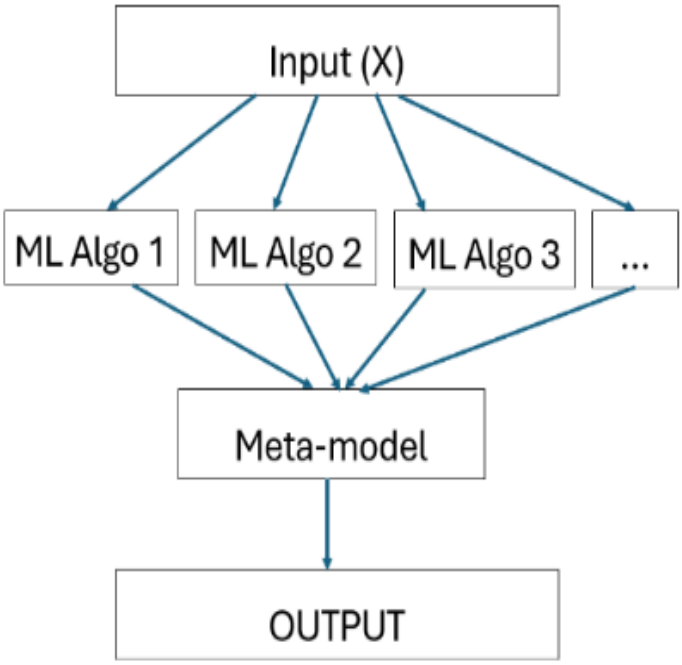


Fig. 3.3. Stacking Process

3.8 Ensemble technique by Bagging

Bagging is applied to each base model, resulting in an ensemble of models that are trained on various training data subsets. Every model used in this procedure is trained on a bootstrapped sample of data, and several instances of each model are involved. The final ensemble forecast is derived from the sum of the predictions made by different models. This scenario applies the bagging technique to Random Forest, SVM, and AdaBoost separately.

This combined strategy, which combines stacking and bagging ensemble methods with node2vec embeddings, highlights a complete methodology for encapsulating the complex interactions seen in the bipartite network of Android functions and apps. The model's prediction accuracy is enhanced by the synergy between ensemble learning algorithms and graph-based embeddings, which also makes the model more resilient and flexible in capturing the complex nature of app similarity.

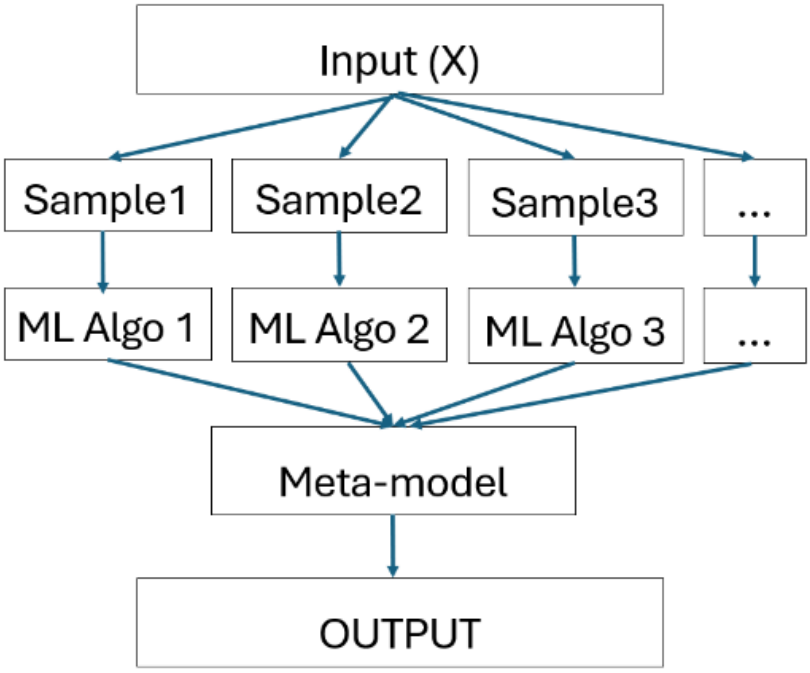


Fig. 3.4. Bagging Process

CHAPTER 4
RESULTS AND DISCUSSION

In this section, first we will discuss about the dataset we have used for our study, CIC-AndMal2017 Android Malware Dataset.

4.1 Dataset

These experiments were performed on the CIC-AndMal2017 dataset which is an extensive compilation of both malicious and safe Android apps that have been hand-picked to give users a more accurate picture of the threat environment on Android smartphones. The dataset uses a novel methodology that runs both malware and benign applications on actual cellphones, in contrast to conventional methodologies. This method seeks to avoid the runtime behavior change used by sophisticated malware strains that have the ability to identify emulation environments.

The collection consists of 10,854 samples in total, of which 6,500 are classified as benign and 4,354 as malware. The safe examples were from the Google Play store and included applications released in 2015, 2016, and 2017. Five thousand samples were chosen, of which 4,26 were malicious and 5,065 were benign, and were installed on real devices in order to verify the dataset's legitimacy and applicability in the real world.

The CIC-AndMal2017 dataset's malware samples are divided into four groups that offer a thorough understanding of the variety of Android threats:

- Adware is malicious software that pretends to be useful apps in order to display intrusive adverts.
- Malware that locks or encrypts a victim's device and demands a payment to unlock it is known as ransomware.
- Scareware is malicious software that deceives users into thinking their device is contaminated and causes them to take action that is not necessary.

- Malicious software that modifies or uses the Short Message Service's (SMS) capability for improper purposes is known as SMS malware.

Table 4.1. Summary of dataset.

Dataset	Type of Applications	Number of Apps	Number of functions	Popular functions
CIC-AndMal2017	Both benign and malicious	#10854	#25685647	#1059767

4.2 Performance evaluation

In this work, we test the effectiveness of the Android malware classifiers using three standard metrics such as: accuracy, F1 score, and AUC [22] [1]. We then go over these metrics in brief [1]. The description outlines a classification system where TN (true negative) signifies the count of accurately predicted benign apps, FP (false positive) denotes benign apps mistakenly classified as malware, TP (true positive) represents the count of accurately predicted malware apps, and FN (false negative) indicates malware apps incorrectly classified as benign. Accuracy (acc.) emerges as the most logical performance metric, representing the proportion of correctly identified applications. [1] .

$$acc = TP+TN / TP+TN+FP+FN.$$

The formula for calculating the F1 score (F1) is:

$$F1 = 2TP / 2TP+FP+FN.$$

In imbalanced datasets, both the accuracy and F1 measurements are deceptive. For example, let's say that just 5% of the applications are harmful. The accuracy of a basic classifier that consistently yields the majority class is 0.95.

In the field of machine learning, the Area Under the Receiver Operating Characteristic Curve (AUC) [22] serves as a performance metric, remaining

insensitive to imbalanced datasets. Regardless of the percentage of malicious applications in the dataset, a perfect classifier will always have an AUC of 1, but a random classifier would always have an AUC close to 0.5. Therefore, when evaluating classifier performance on datasets with varying percentages of dangerous applications, the AUC [22] is the most crucial factor to consider.

4.3 Results

4.3.1 Results of Machine Learning Models

On the test dataset, the SVM model had an accuracy of 86%, meaning that 86% of the instances were correctly identified. This points to a generally solid performance.

With an accuracy of 80%, the XGBoost model meant that 80% of the cases were properly identified. Even though this accuracy is noteworthy, more research is necessary to fully comprehend the behavior of the model.

With an accuracy of 89%, AdaBoost demonstrated strong performance and a high percentage of accurate classifications on the test data.

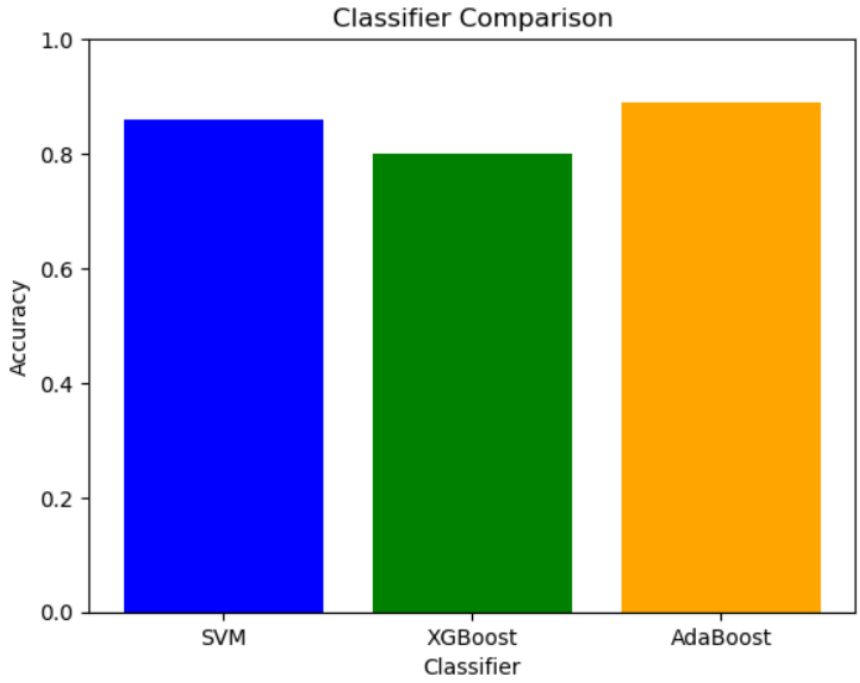


Fig. 4.1. Accuracy result of all three classifier

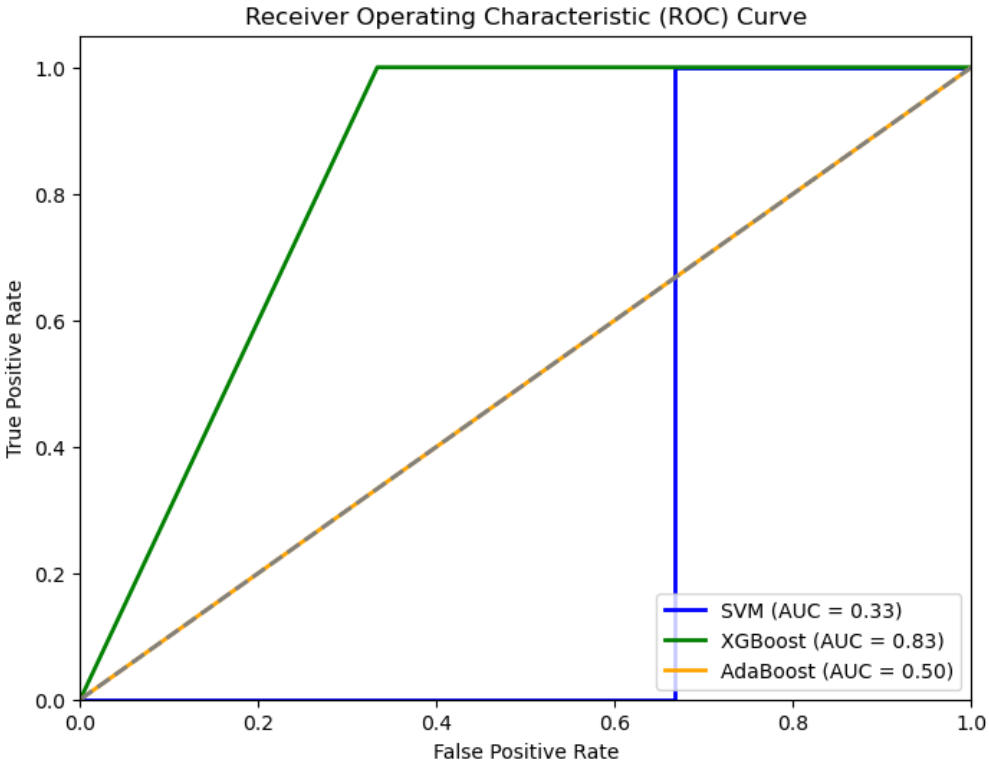


Fig. 5.2. ROC Curve for all three classifiers.

It is discovered that the SVM model's ability to distinguish between classes, as measured by the AUC score, was 0.33. A score of 0.33 indicates that the model's ability to discriminate between classes is restricted and verges on chance.

With an AUC of 0.83, XGBoost has a far higher score. With larger values closer to 1, the model performs better, indicating a strong capacity to differentiate between classes.

AdaBoost's AUC score was a startling 0.5. This implies that the model's class distinction abilities are no more accurate than chance. To determine the cause of this unanticipated outcome, more research is required.

Table 4.2. Summary of Test Accuracies of Three Machine Learning Models.

Model	Test accuracy
SVM Classifier	86%
XG Boost	80%
ADA Boost	89%

4.3.2 Results of Ensemble Models

The node2vec embeddings were used to train each model, and the Max Voting, Stacking, and Bagging techniques were used to combine their predictions.

In this work, we test the effectiveness of the Android malware ensemble classifiers using standard metrics that is accuracy.

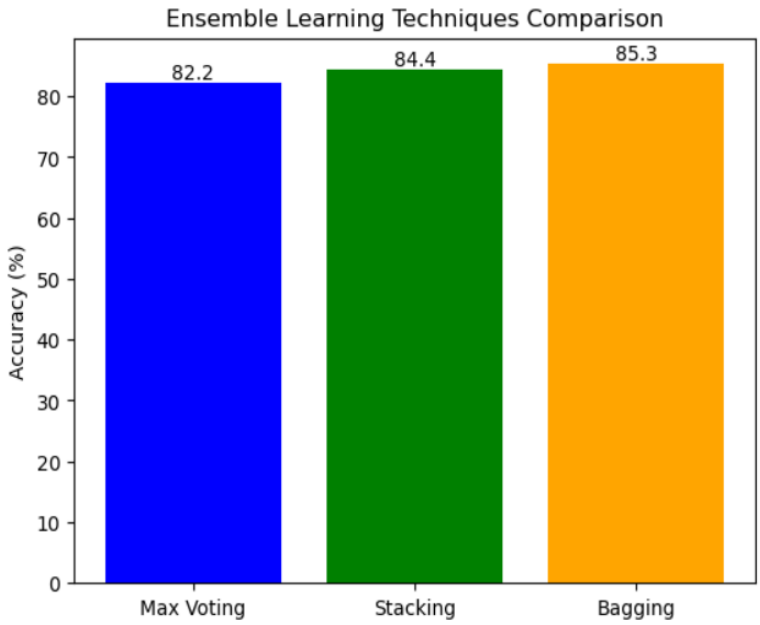


Fig. 4.3. Accuracy result of all three ensemble learning techniques.

The assessment of the group models showed impressive performances, offering information on how well the selected methods worked. Max Voting worked admirably with an accuracy of 82.2%, Bagging was better with an accuracy of 85.3%, and Stacking was noteworthy with an accuracy of 84.4%.

Table 4.3. Summary of Test Accuracies of Three Ensemble Learning Models.

Model	Test accuracy
Ensemble technique by MaxVoting	82.2%
Ensemble technique by Stacking	84.4%
Ensemble technique by Bagging	85.3%

CHAPTER 5
CONCLUSION

This study utilized apk function analysis, node2vec embeddings, and ensemble learning to analyze the links between Android apps (APKs) and their related functions. The process involved extracting a list of distinct APKs from a directory, extracting the functions connected to each APK, and creating a bipartite graph representing the associations. The node2vec technique was used to create node embeddings, capturing the structural details of the graph. Cosine Similarity's similarity ratings were used to quantify app similarities based on common functionalities, providing valuable insights into app relationships. The App Similarity Graph (ASG) was constructed using similarity scores across applications based on their function sets, making it easier to identify related apps.

The group models evaluated performed well, with Max Voting having an impressive 82.2% accuracy rate, bagging outperforming it with an even greater accuracy of 85.3% and stacking with an accuracy of 84.4%. The study highlights the potential of ensemble learning techniques in improving predictive performance.

The study evaluated the performance of machine learning models, specifically Support Vector Machine (SVM), XGBoost, and AdaBoost, in classifying apps into malicious and benign categories using Node2Vec embeddings. The models showed varying levels of accuracy, with SVM achieving 86%, XGBoost achieving 80%, and AdaBoost achieving 89%. However, the AUC scores provided insights into their discriminative abilities, with XGBoost demonstrating a high AUC score of 0.83, indicating strong class separation. The discrepancy between accuracy and AUC scores for SVM and AdaBoost suggests further investigation into their behavior and training challenges. Future analysis should include examination of feature importance, confusion matrices, and precision-recall curves to gain a comprehensive understanding of model behavior. XGBoost emerged as the most promising model, and further fine-tuning and optimization of hyperparameters could enhance its performance.

FUTURE WORK

The categorization of applications will be part of future development, and whether or not the apps are harmful or benign will be determined by the similarity score and app similarity graph. Models may be created utilizing the bipartite graph, similarity scores, and other pertinent information to categorize applications into harmful and benign categories with the use of machine learning techniques. The automatic identification and categorization of potentially hazardous apps using this technology can assist to increase user security.

REFERENCES

[1] T. Frenklach, D. Cohen, A. Shabtai and R. Puzis, "Android malware detection via an app similarity graph," Computers & Security, vol. 109, p. 102386, 2021.

[2] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. Clausen, O. Kiraz, K. A. Yuksel, S. A. Camtepe and S. Albayrak, "Static Analysis of Executables for Collaborative Malware Detection on Android," in 2009 IEEE International Conference on Communications, 2009.

[3] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck and C. E. R. T. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket.," in Ndss, 2014.

[4] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee and K.-P. Wu, "DroidMat: Android Malware Detection through Manifest and API Calls Tracing," in 2012 Seventh Asia Joint Conference on Information Security, 2012.

[5] Y. Aafer, W. Du and H. Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android," in Security and Privacy in Communication Networks, Cham, 2013.

[6] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto and L. Cavallaro, "DroidSieve: Fast and Accurate Classification of Obfuscated Android Malware," in Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, New York, NY, USA, 2017.

[7] H.-J. Zhu, Z.-H. You, Z.-X. Zhu, W.-L. Shi, X. Chen and L. Cheng, "DroidDet: Effective and robust detection of android malware using static analysis along with rotation forest model," Neurocomputing, vol. 272, pp. 638-646, 2018.

[8] M. Bierma, E. Gustafson, J. Erickson, D. Fritz and Y. R. Choe, Andlantis: Large-scale Android Dynamic Analysis, 2014.

[9] M. Y. Wong and D. Lie, "IntelliDroid: A Targeted Input Generator for the

Dynamic Analysis of Android Malware," in #NDSS16#, 2016.

[10] A. Amin, A. Eldessouki, M. T. Magdy, N. Abdeen, H. Hindy and I. Hegazy, "AndroShield: Automated Android Applications Vulnerability Detection, a Hybrid Static and Dynamic Analysis Approach," Information, vol. 10, 2019.

[11] H. Bai, N. Xie, X. Di and Q. Ye, "FAMD: A Fast Multifeature Android Malware Detection Framework, Design, and Implementation," IEEE Access, vol. 8, pp. 194729-194740, January 2020.

[12] W. Niu, R. Cao, X. Zhang, K. Ding, K. Zhang and T. Li, "OpCode-Level Function Call Graph Based Android Malware Classification Using Deep Learning," Sensors, vol. 20, 2020.

[13] Q.-D. Ngo, H.-T. Nguyen, H.-A. Tran, N.-A. Pham and X.-H. Dang, "Toward an approach using graph-theoretic for IoT botnet detection," in Proceedings of the 2021 2nd International Conference on Computing, Networks and Internet of Things, New York, NY, USA, 2021.

[14] R. Taheri, M. Ghahramani, R. Javidan, M. Shojafar, Z. Pooranian and M. Conti, "Similarity-based Android malware detection using Hamming distance of static binary features," Future Generation Computer Systems, vol. 105, pp. 230-247, 2020.

[15] A. Pektaş and T. Acarman, "Deep learning for effective Android malware detection using API call graph embeddings," Soft Computing, vol. 24, p. 1027–1043, 2020.

[16] A. Narayanan, C. Soh, L. Chen, Y. Liu and L. Wang, "Apk2vec: Semi-Supervised Multi-view Representation Learning for Profiling Android Applications," in 2018 IEEE International Conference on Data Mining (ICDM), 2018.

[17] H. Jiang, T. Turki and J. T. L. Wang, "DLGraph: Malware Detection Using Deep Learning and Graph Embedding," in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018.

[18] B. Rashidi, C. Fung and E. Bertino, "Android malicious application detection using support vector machine and active learning," in 2017 13th

International Conference on Network and Service Management (CNSM), 2017.

[19] B. Kang, S. Y. Yerima, K. Mclaughlin and S. Sezer, "N-opcode analysis for android malware classification and categorization," in 2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security), 2016.

[20] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2016.

[21] T. Mikolov, K. Chen, G. Corrado and J. Dean, Efficient Estimation of Word Representations in Vector Space, 2013.

[22] I. H. Witten, E. Frank, M. A. Hall, C. J. Pal and M. Data, "Practical machine learning tools and techniques," in Data mining, 2005.

[23] M. Bierma, E. Gustafson, J. Erickson, D. Fritz and Y. R. Choe, *Andlantis: Large-scale Android Dynamic Analysis,* 2014.

[24] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. Clausen, O. Kiraz, K. A. Yuksel, S. A. Camtepe and S. Albayrak, "Static analysis of executables for collaborative malware detection on android," in *2009 IEEE International Conference on Communications*, 2009.

[25] W. Z. Zarni Aung, “Permission-based android malware detection,” International Journal of Scientific & Technology Research, vol. 2, p. 228–234, 2013.

[26] M. K. Alzaylaee, S. Y. Yerima and S. Sezer, “DL-Droid: Deep learning based android malware detection using real devices,” Computers & Security, vol. 89, p. 101663, 2020.

[27] M. Alazab, M. Alazab, A. Shalaginov, A. Mesleh and A. Awajan, “Intelligent mobile malware detection using permission requests and API calls,” Future Generation Computer Systems, vol. 107, pp. 509-521, 2020.

[28] Y. Du, J. Wang and Q. Li, “An Android Malware Detection Approach

Using Community Structures of Weighted Function Call Graphs," IEEE Access, vol. 5, pp. 17478-17486, 2017.

[29] D. Chaulagain, P. Poudel, P. Pathak, S. Roy, D. Caragea, G. Liu and X. Ou, "Hybrid Analysis of Android Apps for Security Vetting using Deep Learning," in 2020 IEEE Conference on Communications and Network Security (CNS), 2020.

[30] J. Sahs and L. Khan, "A Machine Learning Approach to Android Malware Detection," in 2012 European Intelligence and Security Informatics Conference, 2012.

[31] W. Yang, X. Xiao, B. Andow, S. Li, T. Xie and W. Enck, "AppContext: Differentiating Malicious and Benign Mobile App Behaviors Using Context," in 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015.

[32] S. Hou, A. Saas, L. Chen and Y. Ye, "Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs," in 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW), 2016.

[33] L. Chen, M. Zhang, C.-y. Yang and R. Sahita, "POSTER: Semi-supervised Classification for Dynamic Android Malware Detection," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 2017.

[34] Z. Xu, K. Ren, S. Qin and F. Craciun, "CDGDroid: Android Malware Detection Based on Deep Learning Using CFG and DFG," in Formal Methods and Software Engineering, Cham, 2018.

PUBLICATIONS

[1] M. K. Sahu and R. Gupta, " Android Malware Detection Using Function Analysis in Conjunction with Ensemble Techniques" 4th International Conference on Machine Learning and Big Data Analytics (ICMLBDA), Kurukshetra, India, 2024.

[2] M. K. Sahu and R. Gupta, " Android Malware Detection Using Graphical Techniques" 5th IEEE India Council International Subsections Conference (INDISCON 2024), Chandigarh, India, 2024.

Gmail **Monish Sahu <99monishsahu@gmail.com>**

Accepted paper in the EquinOCS system

EquinOCS <equinocs-admins@springernature.com> 28 March 2024 at 16:11
To: Monish Sahu <99monishsahu@gmail.com>

This message has been sent by the EquinOCS system
https://equinocs.springernature.com/

PLEASE DO NOT REPLY

========

Dear Monish Sahu,

We are pleased to inform you that your paper

060: "Android Malware Detection Using Function Analysis in Conjunction with Ensemble Techniques"

has been accepted for

4th International Conference on Machine Learning and Big Data Analytics (ICMLBDA2024)

Please find the reports beneath.

=== Comment ===

Congratulations! Your submitted paper has been reviewed and accepted for presentation at the IAASSE Technically Sponsored 4th International Conference on Machine Learning and Big Data Analytics (ICMLBDA 2024), to be held from May 9-11, 2024, in National Institute of Technology kurukshetra.

All accepted registered and presented papers will be submitted for possible inclusion Springer Proceedings in Mathematics & Statistics. (Format Available at following link: https://www.springer.com/kr/authors-editors/conference-proceedings/conference-proceedings-guidelines)

1. Register (transfer online paper registration fee) your accepted paper to the conference by following instructions of registration available at the conference website (Early Bird Registration Date: 10 April, 2024) : Registration Details Available at: https://icmlbda2024.iaasse.org/register.html

2. Please strictly follow the Springer Template and revise as per REVIEWERS' COMMENTS for your paper, which are intended to help you to improve your paper before the final publication. The listed comments should be addressed carefully in your revision.

3. Your final paper MUST be submitted by April 08, 2024.

4. In order for your paper to be published in the ICMLBDA- 2024 conference proceedings, ensure the following checklist before Camera Ready Paper submission:

i) Organization of the paper follows; title, abstract, keywords, introduction, related work, proposed work, result analysis, conclusion, and references.
ii) The article must be free from typographical errors which may be carefully looked at.
iii) Check the Author names, affiliation details, Image & Table quality in the paper.
iv) Minimum 12 references are expected and all references must be cited in the paper, citation format should be [1],

09-11 May 2024

Springer

4th International Conference on

Machine Learning and Big Data Analytics (ICMLBDA) 2024 (Hybrid Mode)

Venue:

Department of Electronics and Communication Engineering
National Institute of Technology, Kurukshetra, Haryana

Important Dates

Special Session Proposals:
Feb 05, 2024

Full Paper Submission :
Feb 29, 2024

Acceptance Notification:
March 25 , 2024

Early Bird Registration:
April 10, 2024

Camera ready Paper:
April 15, 2024

ABOUT CONFERENCE

ICMLBDA 2024 provides a platform for researchers and professionals to share their research and reports of new technologies and applications in ML and Big Data Analytics like biometric recognition systems, medical diagnosis, industries, telecommunications, AI Petri Nets model-based diagnosis, gaming, stock trading, intelligent aerospace systems, robot control, law, remote sensing and scientific discovery agents and multiage systems, and natural language and Web intelligence. The conference program will include special sessions, presentations delivered by researchers from the international community, including presentations from keynote speakers and state-of-the-art lectures.
The ICMLBDA aims to bridge the gap between these non-coherent disciplines of knowledge and fosters unified development in next generation computational models for machine intelligence.

Chief Patron

Dr. B.V. Ramana Reddy
Director, NIT Kurukshetra

Patron

Dr. Brahmjit Singh
Professor ECE Dept. NIT Kurukshetra

Co-Patron

Mr. Karan Sharma
HoD ECE Dept. NIT Kurukshetra

Organizing Chair(s)

Dr. Pankaj Verma
Dr. Chhagan Charan
ECE, NIT Kurukshetra
Dr. Mohit Dua
CoE, NIT Kurukshetra

Organizing Secretary(s)

Dr. T N Sasamal
ECE, NIT Kurukshetra
Dr. Ankit Kumar Jain
CoE, NIT Kurukshetra

Conference Tracks

Track 1: Machine Learning
Track 2: Big Data Analytics

Submission Guidelines

Papers reporting original and unpublished research results pertaining to the related topics are solicited.
Full paper manuscripts must be in English of up to 10 pages as per Springer format.

PAPER SUBMISSION

Indexing

ALL ACCEPTED & PRESENTED papers will be published in SCOPUS indexed Springer Proceedings in Mathematics & Statistics (PROMS).
ISI Conference Proceedings Citation Index - ISI Web of Science, Scopus, DBLP.

https://icmlbda2024.iaasse.org/

All accepted and Presented Papers will be published in Springer Book Series

June 05, 2024

Publication (Tentative)

December 2024

SUBMIT PAPER NOW

Indexing

Post conference, proceedings will be made available to the following indexing services for possible inclusion:

Scopus®

dblp computer science bibliography

Clarivate Analytics

Google Scholar

Conference Tracks

Receipt from I.A.A.S.S.E.

Receipt #1632-2056

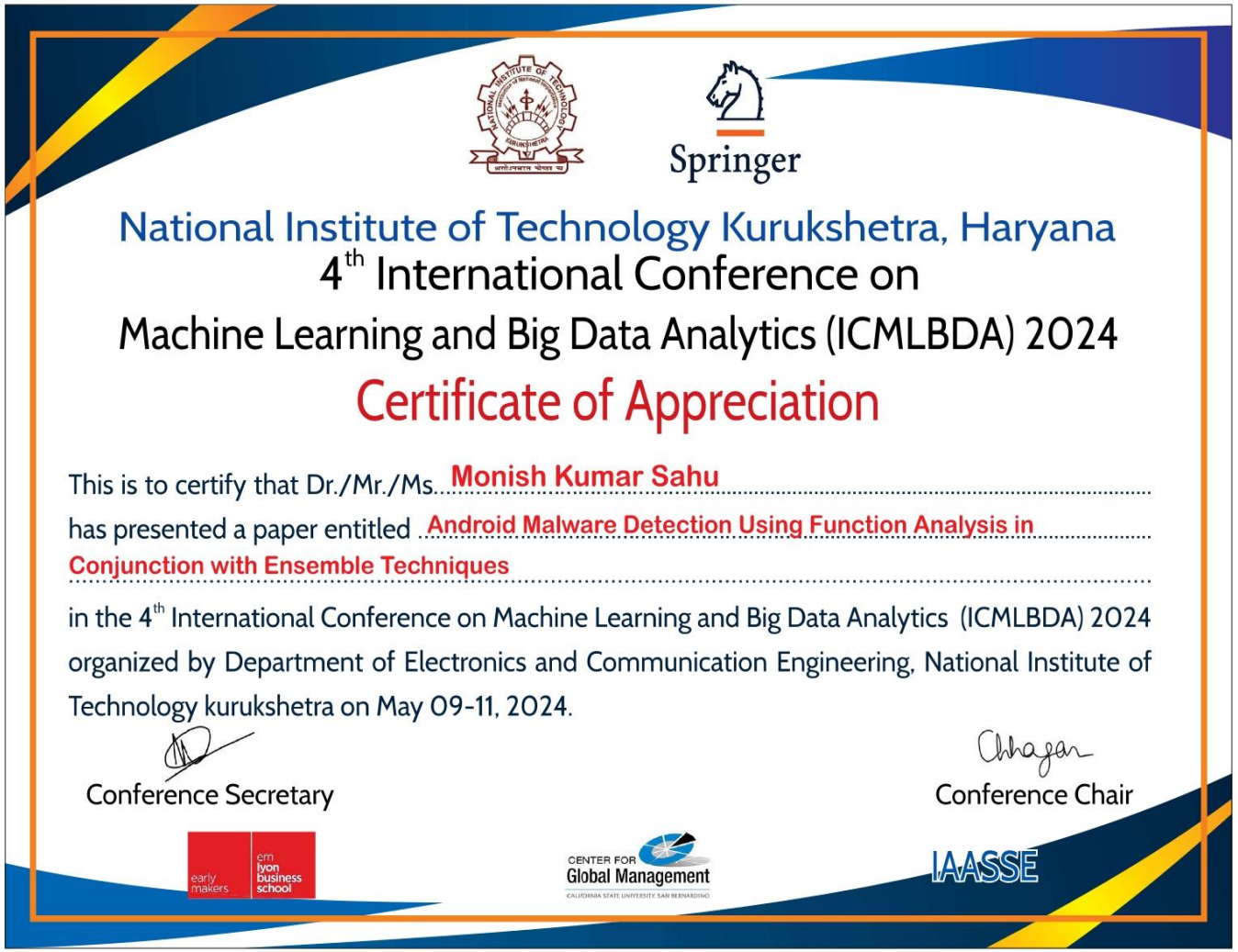
AMOUNT PAID	DATE PAID	PAYMENT METHOD
₹8,000.00	Apr 10, 2024, 12:34:24 PM	VISA - 4753

SUMMARY

Full Time students (Non IAASSE Members) × 1	₹8,000.00
Amount charged	**₹8,000.00**

Paper Title: Android Malware Detection Using Function Analysis in Conjunction with Ensemble Techniques
Paper ID: 060

If you have any questions, visit our support site at https://isms2023.iaasse.org/contact.html or contact us at info@iaasse.org.


Springer
National Institute of Technology Kurukshetra, Haryana
4th International Conference on
Machine Learning and Big Data Analytics (ICMLBDA) 2024
Certificate of Appreciation
This is to certify that Dr./Mr./Ms. Monish Kumar Sahu
has presented a paper entitled Android Malware Detection Using Function Analysis in Conjunction with Ensemble Techniques
in the 4th International Conference on Machine Learning and Big Data Analytics (ICMLBDA) 2024 organized by Department of Electronics and Communication Engineering, National Institute of Technology kurukshetra on May 09-11, 2024.
Conference Secretary
Conference Chair
early makers
em lyon business school
CENTER FOR Global Management
IAASSE

Gmail **Monish Sahu <99monishsahu@gmail.com>**

Decision on Paper ID 2080 of INDISCON 2024.

Microsoft CMT <email@msr-cmt.org> 7 June 2024 at 21:41
Reply-To: Arun Kumar Singh <ieeeindiscon2024@gmail.com>
To: Monish Kumar Sahu <99monishsahu@gmail.com>

Dear Author,

We are pleased to inform you that your Paper ID 2080 Titled "Android Malware Detection Using Graphical Technique" has been accepted for oral/poster presentation at the INDISCON 2024. Further details can be found on conference website (https://ieeeindiscon.org). The detailed reviews/comments given by the reviewers are available in your Microsoft CMT Account.

Please consider the comments provided by the reviewers and revise your paper based on the comments.

Please note the following:
1. At least one of the authors of every accepted paper must register for the conference as author and present the paper in order for it to be included in the conference proceedings of INDISCON 2024, and subsequent submission to IEEE Xplore digital library.
2. For the paper to be accepted in the Final programme, it is expected that at least one author is registered, and Camera Ready Paper is submitted. Non-presented papers will not be submitted to IEEE Xplore digital library as per IEEE no-show policy.
3. Papers presented in the Conference will be eligible for submission for further consideration of publishing in the IEEE Xplore, subject to maintenance of quality, and post-conference scrutiny of response of Conference Organizers to Technical Program of Questionnaire (TPQ) on the Conference.
4. Instructions for submission of Camera Ready Paper will be notified shortly on conference website.

Looking forward to seeing you at INDISCON 2024 at PEC Chandigarh.

Best Wishes,
TPC Chairs, INDISCON 2024

CALL FOR PAPERS [EXTENDED DEADLINE]

5th IEEE India Council International Subsections Conference
INDISCON 2024

August 22-24, 2024,

Venue- Punjab Engineering College (Deemed to be University), Chandigarh, India

Theme- Science, Technology and Society

Patron:
Prof. Baldev Setia, *Director PEC Chandigarh*
Prof. Debabrata Das, *Chair IEEE India Council*

Co-Patron:
Prof. Prerna Gaur, *Chair-Elect, IEEE India Council*
Prof. A. Q. Ansari, *Chair, IEEE Delhi Section*

Honorary Chair:
Prof. Rudra Pratap, *VC Plaksha University*
Prof. Lalit Awasthi, *Director NIT UK*
Prof. Anupam Shukla, *Director NIT Surat*
Prof. B. K. Panigrahi, *IIT Delhi*
Mr. B. A. Sawale, *DG, CPRI, Bengaluru*

General Chair:
Prof. Arun Kumar Singh, *PEC Chandigarh*
Prof. Manish Hooda, *SCL Mohali*
Dr. Puneet Mishra, *URSC Bengaluru*

Organizing Secretary:
Dr. Padmavati, *PEC Chandigarh*
Dr. Simranjit Singh, *PEC Chandigarh*
Dr. Manohar Singh, *PEC Chandigarh*

TPC Chair:
Prof. Sudeb Das Gupta, *IIT Roorkee*
Prof. N. S. Chaudhari, *IIT Indore*
Prof. G. Bhuvaneswari, *Mahindra University*

Finance Chair:
Dr. Vijayalata Yellasiri, *Treasurer, IEEE*
Dr. Sneha Kabra, *Delhi University*
Mr. Mayank Gupta, *PEC Chandigarh*

Publication Chair:
Prof. Jawar Singh, *IIT Patna*
Prof. Jagdish Kumar, *PEC Chandigarh*
Prof. Balwinder Raj, *NIT Jalandhar*

Executive Steering Committee:
Sh. Deepak Mathur, *2024 IEEE VP MGA*
Prof. Preeti Bajaj, *VC-SA IEEE India Council*
Dr. K. R. Suresh Nair, *IEEE India Council*
Prof. M. N. Hooda, VC *IEEE Delhi Section*
**other committees are mentioned on the conference website.*

About the Conference

INDISCON is the flagship International Conference organised by IEEE India Council and IEEE Subsections in India to bring together researchers from academia and industries on various aspects of Sciences, Engineering and Technology. The Conference provides an excellent international platform for sharing of state-of-the-art research/technologies in the field of Electronics, Electrical, Information Technology etc., wherein many national/international eminent personalities will share their vision, expertise and knowledge.

INDISCON 2024 is organised by IEEE Chandigarh Subsection and hosted by Punjab Engineering College (Deemed to be University), Chandigarh along with IEEE India Council. INDISCON 2024 will include a wide range of technical sessions, invited talks, workshops, tutorials, special sessions, industry sessions, exhibits etc.

Technical Tracks

Track 1: Power and Energy Systems
Track 2: Power Electronics, Drives and Intelligent Control
Track 3: Instrumentation, Control and Signal processing
Track 4: Artificial Intelligence and Data Science
Track 5: Communication, Networks & IOT
Track 6: Next Generation Computing and applications
Track 7: Security & Privacy
Track 8: RF/Microwave/Terahertz Technologies
Track 9: Semiconductor Devices
Track 10: VLSI & Embedded Systems
Track 11 : Nanotechnology Materials and Devices
Track 12 : Education Technologies
Track 13 : Women in Engineering

Submission Guidelines

Paper submission instructions and template will be available at http://ieeeindiscon.org/
Paper submission link: https://cmt3.research.microsoft.com/INDISCON2024/

Papers (upto 6 pages in .pdf) presented in the Conference, duly accepted after peer review, will be eligible for submission for further consideration of publishing in the IEEE Xplore, subject to maintenance of quality, and post-conference scrutiny of response of Conference Organizers to Technical Program of Questionnaire (TPQ) on the Conference.

Note: Based on the significance of the work, novelty and technical contents, papers will be selected for the Best Poster Award and Best Paper Award. Travel grant will be awarded to a limited number of applicants on a highly competitive basis, for more details visit conference website.

Important Dates

• Last Date of Paper Submission **[Extended]**	**April 15, 2024**
• Notification of Acceptance	May 15, 2024
• Camera Ready/Final Paper Submission	June 10, 2024
• Last Date of Registration	June 15, 2024

+91-7814171121 ieeeindiscon2024@gmail.com http://ieeeindiscon.org/

About us

INDISCON is a flagship annual international conference of the IEEE India Council organized by an IEEE Subsection in INDIA. INDISCON 2024 scheduled during **August 22-24, 2024**, is being organized by IEEE Chandigarh Subsection along with IEEE India Council. The conference will be hosted by **Punjab Engineering College (Deemed to be University), Chandigarh**. The conference aims to provide an interdisciplinary platform for the academicians, researchers, industry professionals and research scholars to exchange and share their knowledge, experience & research.
Proceedings of previous versions of the conference are available here

Previous Edition	Dates	Venue	Theme
IEEE INDISCON 2023	August 5-7, 2023	GSSS Institute of Engineering & Technology for Women, Mysuru	Computational Intelligence and Learning Systems
IEEE INDISCON 2022	July 15-17, 2022	KIIT Deemed to be University, Bhubaneswar	Impactful Innovations for Benefits of Society and Industry
IEEE INDISCON 2021	August 27-29, 2021	Visvesvaraya National Institute of Technology, Nagpur	Impactful innovations for the benefit of industry and society
IEEE INDISCON 2020	October 3-4, 2020	Gayatri Vidya Parishad College of Engineering, Visakhapatnam	Smart and Sustainable Systems - Decade Ahead

IEEE India Council

IEEE is the world's largest professional association dedicated to advancing technological innovation and excellence for the benefit of humanity. IEEE and its members inspire a global community through IEEE's highly cited publications, conferences, technology standards, and professional & educational activities. IEEE India Council is the umbrella organisation which coordinates IEEE activities in India. Its primary aim is to assist and coordinate the activities of local "Sections", in order to benefit mutually, and avoid duplication of effort and resources. IEEE India Council was established on May 20, 1976 and is one of the five councils in the Asia Pacific Region (Region #10 of IEEE).
Details

IEEE Chandigarh Subsection

IEEE Chandigarh Subsection is a technical society that was established on June 18, 2005, under IEEE Delhi Section of IEEE India Council. It provides a platform for the students to enhance their technical skills and professional growth. The subsection organizes various events and technical extravaganzas, such as Techadroit, which is an annual event organized by IEEE PEC Student Branch in association with IEEE Chandigarh Subsection for students. In 2020, the subsection organized the first-ever Chandigarh Subsection Congress with the participation of more than 1700 delegates.
Details

ticketdude.

INVOICE

Place of supply: Karnataka
NO: 42b09d66-573b-4752-bdee-31d82c65842c

5TH IEEE INDIA COUNCIL INTERNATIONAL SUBSECTIONS CONFERENCE 1 (INDISCON 2024)
IEEE India Council, Bangalore & IEEE Chandigarh Sub-section, Chandigarh

---

Date Issued: 2024-06-29
Invoice to: **Monish Kumar Sahu**

DESCRIPTION	QUANTITY	AMOUNT
Event : IEEE INDISCON 2024 Early Bird Non-IEEE Student Members (Indian)	1	INR 6000.00
Discount		INR 0.00
Processing Fee		INR 300.00
Surcharge Fee		INR 0.00
GRAND TOTAL		**INR 6300.00**

Six Thousand, Three Hundred Rupees, Zero Paise

---

*All Currency in INR

This is a system generated invoice and does not need any signature. Thank you

ANNEXURE-IV

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahbad Daulatpur, Main Bawana Road, Delhi-42

PLAGIARISM VERIFICATION

Title of the Thesis ANDROID MALWARE DETECTION USING GRAPHICAL TECHNIQUES Total Pages 34 Name of the Scholar Monish Kumar Sahu
Supervisor (s)

(1) Mr. Rahul Gupta

(2)_____

(3)_____

Department INFORMATION TECHNOLOGY, DELHI TECHNOLOGICAL UNIVERSITY

This is to report that the above thesis was scanned for similarity detection. Process and outcome is given below:

Software used: Turnitin Similarity Index: 14% , Total Word Count: 6981

Date: 30/05/2024

Candidate's Signature **Signature of Supervisor(s)**

PAPER NAME
MonishThesis.pdf

AUTHOR
Monish Sahu

WORD COUNT
6985 Words

CHARACTER COUNT
40450 Characters

PAGE COUNT
34 Pages

FILE SIZE
610.6KB

SUBMISSION DATE
May 31, 2024 9:36 AM GMT+5:30

REPORT DATE
May 31, 2024 9:37 AM GMT+5:30

14% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 7% Internet database
- 10% Publications database
- Crossref database
- Crossref Posted Content database
- 8% Submitted Works database

Excluded from Similarity Report

- Bibliographic material
- Small Matches (Less then 8 words)