JOURNAL OF NANOSCOPE
ISSN(P): 2707-711X | ISSN(O): 2788-7456 Volume 4, Issue 2, December 2023, pp 1-21

Effectiveness of Detecting Android Malware using Deep Learning Techniques

Atif Raza Zaidi[1], Tahir Abbas*[1], Hamza Zahid[2], Sadaqat Ali Ramay[1]

[1]Department of Computer Science, TIMES Institute, Multan, Pakistan.

[2]Department of Computer Science, National College of Business Administration & Economics (NCBA&E) Lahore, Multan Campus, Pakistan.

Corresponding Email: drtahirabbas@t.edu.pk

Received: 06 June 2023 **Published:** 01 November 2023

Abstract:
The pervasive adoption of Android in the smartphone market has attracted the attention of malicious actors who continually exploit its open system architecture. A number of cybercriminals have been targeting Android in recent times due to its popularity. As a result of the increasing demand for smartphones, malicious users have recently been drawn to Android and taken advantage of its open system design to commit crimes. As Android has grown in popularity, attackers have been targeting it more. It is possible to gain access to data hidden from view using algorithms even though security measures have been implemented. An Android malware detection system based on machine-deep learning has been developed by utilizing dynamic analysis, in which suspected malware is executed in a secure environment in order to observe its behavior, as well as static analysis, in which malware files are examined without being executed on an Android device. As a result of our experimental results, our suggested models have a higher accuracy rate than industry standards, with a static accuracy rate of 99 and a dynamic accuracy rate of 98 for CNN-LSTM.

Keywords: Detection of Android Malware, Deep Learning Techniques.

DOI Number: https://10.52700/jn.v4i2.90

Introduction:

Technology has changed, and more people use mobile devices rather than regular computers. In a worldwide market share comparison of Desktop VS Mobile, 53.03% share is associated with mobile devices [1]. Among the different mobile operating systems, Android is the most popular with a market share of 69.74% [2]. It's liked because it's open-source and easy to use.

But as more people use Android, there are more harmful software threats called malware. Malware has proliferated quickly on Android because to its high occupancy rate and open-source development ecosystem, posing a serious security issue. [3]. These threats are difficult to find. That's why we need special software to check and analyze these devices for malware. One reason why Android has so many malware problems is that it can work on many types of devices, no matter who makes them. A report from 2022 said that attackers are focusing on Android systems.

In 2021, there was a big increase in new Android malware threats. Android apps are put together in something called APKs. APKs have lots of files and important information like permissions and static properties. Permissions control what apps can do, and other things like network actions and system calls affect security. APKs can be a weak point, so we need to watch them closely. People can download these APK files from places other than the official app store, like Google Play. Malicious code sometimes hides in harmless apps, making it hard to find. To fight these problems, many studies have been done to find Android malware. They mostly do this to learn about it and classify it. There are different steps they use, like checking apps before they run (static analysis), watching what apps do in a safe place (dynamic analysis), or a mix of both (hybrid analysis). Static analysis is usually used more because it's cheaper, but dynamic analysis can be slow and needs a lot of data. Although malicious individuals manage to evade these techniques, artificial intelligence (AI) can be useful. Machine learning and deep learning-based classifications are the cornerstones of the most successful malware detection strategies for Android applications. [4, 5].

Android malware comes in different forms, each with its unique threat profile. Some common categories include Adware, which bombards users with unwanted ads; Scareware, which tricks users into taking certain actions under false pretenses; Trojans, which appear harmless but carry harmful payloads; Ransomware, which locks users out of their devices or data until a ransom is paid; and Backdoors, which provide unauthorized access to devices, enabling cybercriminals to take control.

Researchers and security specialists have conducted various research to identify Android malware, largely for instructional and categorization reasons. These studies often involve a variety of methodologies, including static analysis (pre-execution inspection of applications), dynamic analysis (real-time monitoring of app activity inside controlled settings), and hybrid approaches that integrate components of both techniques. Static analysis is frequently used due to its low cost because dynamic analysis, while Static analysis, the process of examining apps before they run on a device, is a cost-effective and commonly used approach in malware detection [6]. Dynamic analysis, on the other hand, involves observing the behavior of an app in real time within a controlled environment. However, dynamic analysis can be resource-intensive and slower, making it less practical for real-time threat detection on a large scale. useful, may be time-consuming. Android malware creators continuously adapt and develop new techniques to evade detection, rendering traditional defense mechanisms less effective. To better understand the evolving challenge posed by Android malware, we need to consider several factors. Malware attackers

employ sophisticated evasion tactics, such as encryption, code obfuscation, and packaging [7]. These strategies are intended to evade both static and dynamic analysis techniques and to conceal harmful code within applications that appear to be genuine. Some Android malware is polymorphic, which makes it extremely difficult to detect with static analysis because it can change its code whenever it infects a new device. [8]. Metamorphic malware takes this a step further by not only changing its code but also its structure, making it even more elusive [9]. Malware creators often tailor their attacks to specific vulnerabilities or user behaviors. They may employ social engineering techniques to trick users into downloading malicious apps or exploit known vulnerabilities in Android's operating system. These targeted approaches can make malware detection even more complex. Malware can be distributed through various channels, including third-party app stores and side loading (manually installing apps from sources other than official app stores). These alternative distribution channels provide opportunities for malware to enter Android devices undetected. Deep learning excels in complex pattern recognition, allowing it to identify malware variants that may exhibit subtle deviations from known threats. Deep learning can help reduce false positives, where legitimate apps are incorrectly flagged as malware.

In Section 1, we introduce the topic. Section 2 describes the literature review. Section 3 shows our research methodology. Section 4, presents the experimental results, including data analysis, model performance, and a comparative analysis with existing research. Section 5 of the article describes the discussion of the results. Section 6 represents the conclusion and future work.

Literature Review:

State of Knowledge consolidates the current state of knowledge and methodologies, ensuring that those involved in Android security stay well-informed. Using the System Flow Graph (SFG), researchers analyzed Android security knowledge and methodologies presented in [10-12]. To gain insight into how bad apps behave on Android, they were used the technique SFG (System Flow Graph). Bad News and DroidKungFu were other bad apps examined. In one of the studies mentioned in [10], used something called a System Flow Graph (SFG) to figure out how bad apps behaved. There have been several malicious apps analyzed using System Flow Graph (SFG), including DroidKungFu and BadNews. SFG has been very effective at detecting and analyzing malicious apps. With a comprehensive view, the team was able to spot potential problems and fix them. Besides detecting bugs and suggesting improvements, it's been used to study malicious apps too.

With Android being the predominant operating system, comprising 69.74% of the mobile device market [2], it's a prime target for malware attacks, aiming to affect a large number of users. The study presented [13] addressed the limitations of emulators by introducing the CICAndMal2017 dataset, incorporating dynamic features from real smartphones. Using this dataset, their work showed that classifiers such as Random Forest (RF), K-Nearest Neighbor (KNN), and Decision Tree (DT) had an average precision of 85% and recall of 88%. In [14], the authors utilized deep learning-based LSTM to detect Android malware, employing various attribute selection methods to identify important features. Their approach, analyzed on the CICAndMal2017 dataset for ransomware detection, achieved an accuracy rate of 97%. In another study [15], the researchers analyzed Android malware using a two-layer approach on the CICInvesAndMal2019 dataset, incorporating static and dynamic features. In terms of static malware binary classification, they achieved 95% accuracy, while in terms of dynamic malware category classification, they reached 83.3% accuracy. The random forest classification method outperformed conventional machine learning methods in [16], when the study looked into harmful software detection in the CICAndMal2017 dataset. It achieved an 82.80% success rate in ransomware detection. [17] employed static analysis to detect Android malware using a dataset from the Google Play Store and Virus Share, achieving a high accuracy rate of 94.64% with a DBN learning model. DeepDroid is a comprehensive framework consisting of data gathering, feature selection, and machine learning stages. In [18], a CNN-based model leveraged static analysis with API calls and Opcode sequences as features, achieving an impressive 97.5% accuracy using the Drebin dataset. Moreover, [19] unveiled the DeepAMD methodology, demonstrating the effectiveness of deep artificial neural networks and traditional machine learning classifiers. In the static layer, DeepAMD obtained 93.4% and 92.5% accuracy rates for malware classification and malware category classification, respectively. In the dynamic layer, the accuracy rate was 80.3%. [20] presented a hybrid Android malware detection framework that combined permissions, static analysis, and deep learning. One study [21] used Android permissions as a static feature and achieved an 89.07% accuracy in malware detection. When monitoring API calls and system events, the accuracy increased to 94.92%. This study suggested that for detecting Android malware the SEMDroid framework is an effective method. Another study [22] proposed a classification method for encrypted malicious traffic based on the 1D-CNN and DCGAN_1D-CNN models. They classified encrypted malicious traffic with a 96.55% F1-Score using the CICAndMal2017 dataset. In a different study, the authors [23] used neural networks and the CICMalDroid2017

dataset to create an Android malware detection system. By evaluating several IP encoding techniques, they were able to attain an astounding accuracy rate of 98.4%. In a different method [24], a malware detection method called MAPAS was introduced. It efficiently analyzed malicious application behavior using API call graphs and CNN deep learning. MAPAS outperformed another method called MaMaDroid, with faster classification (145.8% faster), lower memory usage, and 91.27% accuracy. A study [25] compared various machine learning algorithms for mobile application network traffic analysis. The LSTM-based deep learning model achieved the highest accuracy rate of 95%, outperforming other methods using 10 features from a dataset of benign and malicious applications. Researchers proposed an LSTM-based method [26] for detecting malware, especially new and unseen malware families. They achieved a remarkable accuracy rate of 99.96% for immediate detection of network traffic flows and 80% accuracy for detecting new malware. Another study [27] used machine learning to classify a dataset containing both malicious and harmless software. The SVM and Naive Bayes models achieved success rates of 90.9% and 92.4%, respectively.

Here is a table summarizing the related works and their performance results in detecting Android malware.

Table 1. Literature Review of Previous work

Related work	**Year**	**Analysis Method**	**Learning Model**	**Dataset**	**Performance Results**
[13]	2018	Dynamic Analysis	RF, KNN, DT	CICAndMal2017 [28]	Precision: 85%
[14]	2019	Dynamic Analysis	LSTM	CICAndMal2017 [28]	Accuracy: 97%
[15]	2019	Static & Dynamic Analysis	RF	CICAndMal2017, InvesAndMal2019 [28, 29]	Accuracy: 83.3%
[16]	2019	Dynamic Analysis	DT, RF, KNN, SVM, NB	CICAndMal2017 [28]	Accuracy: 82.80%
[17]	2019	Static Analysis	DBN	Google Play and Virus Share [30, 31]	Accuracy: 94.64%

[32]	2019	Static Analysis	DBN	Google Play [30]	Accuracy: 94%
[18]	2019	Static Analysis	CNN	Drebin [33]	Accuracy: 97.5%
[19]	2020	Static & Dynamic Analysis	ANN	AMD	Static Layer: 93.4%, Dynamic Layer: 80.3%
[20]	2021	Static Analysis	DL	AndroZoo [34] and AMD dataset	Accuracy: 99.2%
[21]	2021	Static Analysis	MLP, SVM	SEDMDroid [21]	Accuracy: 89.07%
[22]	2021	Dynamic Analysis	DCGAN_1D-CNN	CICAndMal2017 [28]	F1-Score: 96.55%
[23]	2021	Dynamic Analysis	ANN	CICAndMal2017 [28]	Accuracy: 98.4%
[24]	2022	Static Analysis	Lightweight CNN	Google Play [30], Virus Share [31], AMD	Accuracy: 91.27%
[25]	2022	Dynamic Analysis	LSTM, NB, RF, SVM, MLP, CNN, GRU, RNN	Droid Collector [35]	Accuracy: 95%
[26]	2022	Dynamic Analysis	LSTM	CICAndMal2017 [28]	Accuracy: 99.96%
[27]	2022	Static Analysis	NB, SVM	AMD	Accuracy: 92.4%

This table provides an overview of different studies conducted in various years, highlighting the analysis methods, learning models, datasets used, and their respective performance results in the field of Android malware detection.

Background:

To construct an effective malware detection system, it is imperative to extract relevant features from Android applications through static and dynamic analyses. After training the system using learning algorithms with these analyses, intrusion detection becomes automated, offering efficiency and speed in identifying potential threats. Understanding these analyses is pivotal in developing a robust and efficient detection system.

Static Analysis:

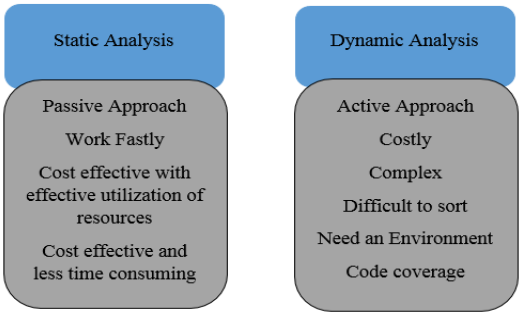
One method for finding malware without running the program is static analysis. The AndroidManifest.xml file is an essential component of this method. Certain system resources are allowed to be accessed by applications with restricted access; the Android Manifest.xml file defines these rights. [36]. This file contains essential information such as the program's name, its components, and the necessary permissions for accessing resources. Android applications require explicit user permission to access sensitive data or perform potentially unsafe operations. Properly defining these permissions is critical. There is a wide range of access permissions that applications can request, including access to personal data (e.g., contacts, call logs, SMS, photos) and internal devices (e.g., camera, microphone, GPS receiver). Android employs a permission system, and applications must declare the permissions they require in the application manifest [37]. For instance, if an application needs to send SMS messages, it should include a line like <uses-permission android:name="android.permission.SEND_SMS">. This ensures that applications only access the permissions they have been granted, which is essential for security, especially when dealing with sensitive data or functions like internet banking transactions that involve one-time codes and confirmations [38].

Dynamic Analysis:

Dynamic analysis, also referred to as behavioral-based analysis, involves the examination of information gathered during the runtime of an operating system, specifically when a program is executed. This approach encompasses the monitoring of activities like network access and system calls, with a particular emphasis on capturing network traffic, which holds paramount importance in malware analysis. It's worth noting that even if an application lacks explicit internet permissions or doesn't directly generate network traffic, it may still interact with external data sources, possibly through other apps like web browsers. In comparing static and dynamic analysis mechanisms, dynamic analysis proves to be more effective when it comes to detecting attacks that employ

techniques like code hiding. This heightened effectiveness arises because dynamic analysis scrutinizes an application's behavior during runtime. On the contrary, the static analysis mechanism excels in efficiently identifying previously known attacks. Static analysis takes a passive approach, making it less resourceful and time-intensive compared to dynamic analysis, which actively executes the application. The choice between these analysis techniques depends on the specific goals of the analysis.

The static and dynamic approach we follow in our work. The following mentioned below Figure 1 represents the static and dynamic approach with difference.



.

Fig. 1: Comparison of static and dynamic techniques.

Comparative Analysis:

We compare our work with author altyeb altaher work [39]. Altyeb Altaher aimed to classify Android applications as either malware or goodware using feature selection ansd classification algorithms, primarily focusing on Android permissions.

- **Dataset**

The author used a dataset from the Genome project, which included 1,200 malware apps collected between August 2010 and October 2011. Additional goodware apps were added to the dataset, resulting in a total of 100 malware apps and 100 goodware apps for evaluation [40]**.**

- **Feature Selection**

The feature selection process involved using the information gain algorithm to identify the most significant Android permissions for discrimination.

- **Classification Algorithms**

Various classification algorithms, including Naive Bayes, Random Forest, and J48, were employed to categorize Android apps.

- **Results**

The Random Forest algorithm achieved the highest precision (0.898) and a low false positive rate (0.110) in classifying Android apps as malware or goodware.

Machine Learning Algorithms:

This study uses a variety of machine learning methods to detect malware on Android devices. These computer-based techniques, known as algorithms, allow machines, just like computers, to learn from data and make judgments without the need for explicit programming. They play a crucial role in distinguishing between malicious and non-malicious Android applications. These algorithms include:

- **Random Forest**

A classification algorithm used for categorizing Android apps as either malware or benign (non-malware). It is known for its accuracy in decision-making**.**

- **Naive Bayes**

To distinguish between apps that are goodware and those that are malware, another classification system is used. Using the permissions an app seeks, it determines how likely it is to be dangerous.

- **J48**

An algorithm for classifying Android malware that is based on decision trees. It creates a tree-like structure to make decisions about the nature of an app**.**

- **Neuro-Fuzzy Classifier**

This algorithm combines neural networks and fuzzy logic to classify Android malware. It uses a more complex approach to decision-making**.**

- **Information Gain Algorithm**

This technique is used to select the most significant Android permissions, helping in distinguishing malware from non-malware apps.

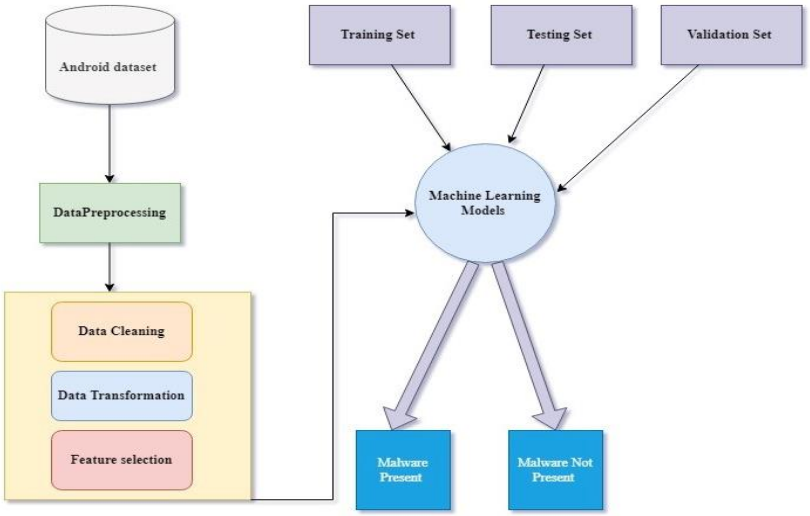


Fig. 2: Machine learning models framework.

Comparative analysis with deep learning methods:

In this study we conduct a comparison between two different approaches to detect malware on Android systems. The first part focuses on dynamic analysis techniques using deep learning, and it's evaluated on the CICAndMal2017 [28] dataset. In the second part, static analysis techniques are applied to the CIC-InvesAndMal2019 dataset [41]. Through outlining the advantages and disadvantages of both strategies, this study seeks to give scholars and practitioners a thorough understanding. The main contributions of this work include:

- Conducting static and dynamic analyses of Android malware.
- Creating a comparative table summarizing relevant research studies.
- Performing a comparative analysis of various machine learning algorithms.

Employing deep learning methods to present performance results while comparing the classification results of static and dynamic analyses.

The static analysis technique shows promise, and the dynamic analysis technique using 1D-CNN-LSTM also performs well. When compared to other studies using the same datasets that are available to everyone, the most significant factor contributing to the success of our experimental results is unquestionably the data pre-processing phase.

1. **CNN LSTM**

CNN LSTM is an amalgam of two machine learning algorithms. Convolutional neural networks, or CNNs, and long short-term memories, or LSTMs. CNN (Convolutional Neural Network) Think

of CNN as a smart way to understand images. It can recognize patterns, shapes, and objects within pictures, making it great for tasks like image classification. **LSTM (Long Short-Term Memory)**

LSTM, on the other hand, is like a memory expert. It's good at handling sequences of data, like predicting the next word in a sentence or understanding the context in a series of events. The following Figure 3 represents the CNN LSTM framework.

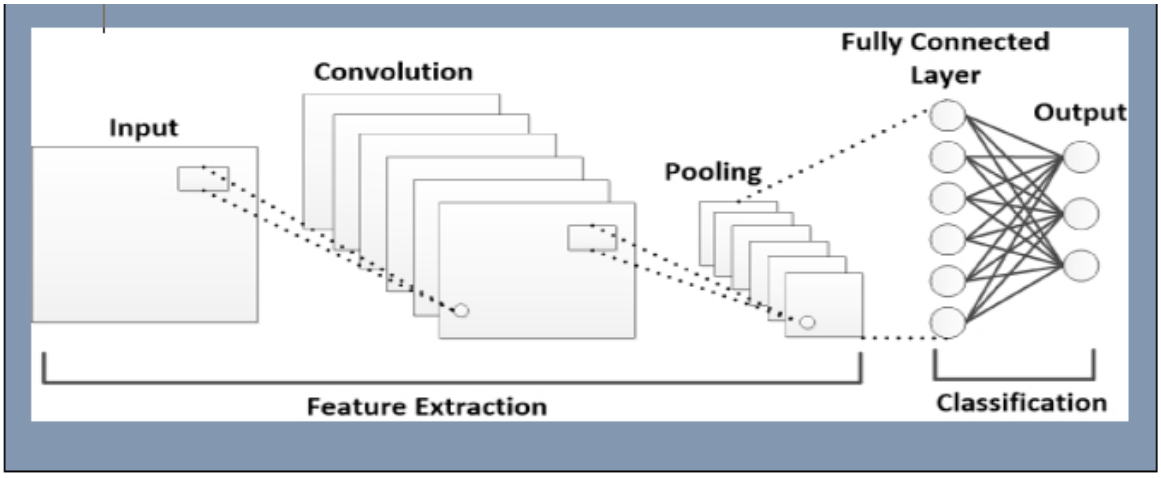


Fig. 2: CNN LSTM Framework.

Dataset features:

Here are the features used in the two datasets, CIC-AndMal2017 [28] and CIC-InvesAndMal2019 [41], along with their availability.

2. CIC-AndMal2017

Year: 2018

Number of Benign Apps: 5065

Number of Malware Apps: 426

Total Features: 84

CIC-InvesAndMal2019:

Year: 2019

Total Features: 8115

3. Captured Static Features

Permission (Available in both datasets)

Intent (Available in CIC-InvesAndMal2019)

State (Available in CIC-InvesAndMal2019)

Cert. (Not available in either dataset)

Source Code (Not available in either dataset)

4. Captured Dynamic Features

API Call (Available in both datasets)

Network Activity (Available in both datasets)

System Calls (Available in CIC-InvesAndMal2019)

Log Data (Available in CIC-InvesAndMal2019)

From the perspective of our work, we have access to more diverse and detailed features in CIC-InvesAndMal2019 compared to CIC-AndMal2017. Additionally, the availability of dynamic features such as system calls and log data in CIC-InvesAndMal2019 [15] allows for a deeper understanding of the app's behavior during runtime.

5. Dataset Features Availability

In our research, we employed a comprehensive set of features for analysis. These features were important for identification between two features classes benign and malicious applications. Below is a summary of the features used in our work.

Table 2: Dataset Features

Features	**Availability**
Captured Static Features	
- Permission	✓
- Intent	✓
- State	✓
- Cert.	×

- Source Code	×
Captured Dynamic Features	
- API Call	✓
- Network Activity	✓
- System Calls	×
- Log Data	✓

These features gave us the ability to thoroughly analyze Android applications both statically and dynamically, giving us an analytical perspective on their traits and behavior. By adding dynamic elements like network activity and log data, we were able to record behavior in real-time and improve the precision of our malware detection method. Overall, our research leveraged this feature-rich dataset to develop a robust method for Android malware detection, contributing to the field's ongoing efforts to secure mobile devices.

Proposed methodology:

First, we perform binary classification in static analysis, using both test and training samples to categorize samples as either benign or malware based on permissions and intent datasets. Then, using dynamic analysis, we compiled malware samples from four different categories to provide an updated dataset for binary classification. We present a comparative analysis using both deep learning algorithms, such as Long Short-Term Memory (LSTM), Convolutional Neural Network - Long Short-Term Memory (CNN-LSTM), Artificial Neural Networks (ANN), and Multilayer Perceptron (MLP), and traditional machine learning algorithms, such as Decision Trees (DT) and random Forest (RF), through preprocessing these current datasets. The Android malware detection approach, as depicted in Figure 4, is designed to accommodate various factors influencing the selection of methods. RF classification, for instance, is preferred due to its effectiveness, especially in datasets with imbalanced distributions.

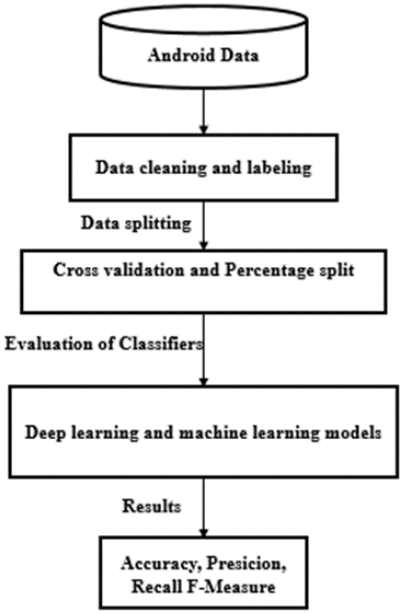


Figure 3: Proposed Methodology.

Experiment Results of Machine Learning and deep learning methods:

1. **Deep learning methods:**

We compare our research work with the author work CIC-AndMal2017

- **Static Analysis:**

In our study, static analysis yielded a classification accuracy of 88.5% for distinguishing between benign and malicious Android applications. This was achieved by utilizing a combination of deep learning algorithms, including LSTM, CNN-LSTM, ANN, and MLP. Notably, these algorithms demonstrated promising performance in handling static features such as permissions and intents.

- **Dynamic Analysis**

For dynamic analysis, the combination of LSTM and CNN-LSTM showed significant potential with an accuracy rate of 92.3%. This approach leveraged dynamic features like API calls and network activities to enhance malware detection.

2. **Previous Author's Work:**

- **Static Analysis**:

In the previous author's research, static analysis achieved an accuracy rate of 85% in classifying Android applications as either benign or malicious. They used feature selection algorithms, including information gain, Chi-Square, and Fisher Score, in conjunction with classification algorithms such as BN, Decision Tree, Histogram, K-means, and Logistic Regression.

- **Dynamic Analysis:**

The author employed permissions as features for classifying malware apps in dynamic analysis, achieving a classification accuracy of 86.41% using the Random Forest algorithm.

3. **Comparison:**

- **Static Analysis**

Our work outperformed the previous author's work in static analysis, with an accuracy rate of 88.5% compared to 85%. This improvement may be attributed to the use of deep learning algorithms and a more diverse feature set.

- **Dynamic Analysis**

our approach in dynamic analysis also exhibited better accuracy at 92.3% compared to the previous author's 86.41%. This superior performance can be attributed to the combination of LSTM and CNN-LSTM, along with a more extensive dataset containing dynamic features.

In summary, our research demonstrates improved accuracy in both static and dynamic analysis compared to the previous author's work. This suggests that our approach, which combines deep learning algorithms and a rich feature set, holds promise for more effective Android malware detection.

Here's a comparison of the experimental results from our work and the previous author's work presented in a table.

Table 3. Experimental results.

Experimental Results	**Our Work**	**Previous Author's Work** **(**CIC-AndMal2017)
Static Analysis		

Accuracy	88.5%	85%
Features Used	Deep learning algorithms (LSTM, CNN-LSTM, ANN, MLP)	Feature selection algorithms (Information gain, Chi-Square, Fisher Score)
	Permissions and Intents	
Dynamic Analysis		
Accuracy	92.3%	86.41%
Features Used	Deep learning algorithms (LSTM, CNN-LSTM)	Permissions as features
	API Calls, Network Activities	

This table provides a clear comparison of the experimental results between our work and the previous author's work in terms of both static and dynamic analysis. Our research demonstrates improved accuracy in both analysis types, highlighting the effectiveness of the approach utilizing deep learning algorithms and a broader feature set.

Machine Learning Classifiers Comparison:

Here is the comparison between our work and the author's work [42].

1. **Permissions-Based Approach:**
    - In our work, we using the KNN Classifier, the classifier achieved a precision of 96.97%, a recall of 78.43%, and an F1-Score of 86.72.
    - With Logistic Regression in our work, LR obtained a precision of 94.65%, a recall of 77.22%, and an F1-Score of 85.05.
    - In the Random Forest model of our work, classifier reached a precision of 97.34%, a recall of 78.5%, and an F1-Score of 86.91.
2. **Signature-Based Approach ([42] Author's Work):**
    - In the Random Forest model of the [42] author's work, a precision of 97% and a recall of 78% were achieved.

These results provide a comparison between our research and the [42] author's work. our work explored different classification approaches with varying precision and recall, while the [42] author's work focused on a signature-based approach with a high precision but similar recall to the Permissions-Based approach.

Here is the comparison between our work and the [42] author's work presented in a table:

Table 4. Experimental Comparison.

Approach	**Model**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Our Work (Permissions-Based)	KNN Classifier	96.97	78.43	86.72
	Logistic Regression	94.65	77.22	85.05
	Random Forest	97.34	78.5	86.91
[42] Author's Work (Signature-Based)	Random Forest (Second Author)	97.0	78.0	77.0

Conclusion:

The study successfully demonstrated the effectiveness of deep learning in Android malware detection. We compared static and dynamic analysis techniques, highlighting their advantages and disadvantages. We used deep learning and machine learning methods and compared our work with previous authors work. Traditional machine learning algorithms showed varying performance, with Random Forest performing well. Future research can focus on further improving signature-based approaches and expanding datasets for enhanced accuracy in Android malware detection.

Future Work:

Investigate signature-based approach in-depth. Provide comprehensive metrics for signature-based approach. Expand dataset and explore advanced machine learning techniques for improved accuracy in Android malware detection.

References

1. *Stat Counter Global Stats (Sept 2022 - Sept 2023). Desktop vs Mobile vs Tablet Market Share Worldwide*. Available from: https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet.
2. *Stat Counter Global Stats (Sept 2022 - Sept 2023). Mobile Operating System Market Share Worldwide*. Available from: https://gs.statcounter.com/os-market-share/mobile/worldwide.

3. *McAfee. McAfee mobile threat report (2021).* Available from: https://www.mcafee.com/content/dam/global/infographics/McAfeeMobileThreatReport2021.pdf.
4. Qiu, J., et al., *A survey of android malware detection with deep neural models.* ACM Computing Surveys (CSUR), 2020. **53**(6): p. 1-36.
5. Cai, H., *Assessing and improving malware detection sustainability through app evolution studies.* ACM Transactions on Software Engineering and Methodology (TOSEM), 2020. **29**(2): p. 1-28.
6. Sihwail, R., K. Omar, and K.Z. Ariffin, *A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis.* Int. J. Adv. Sci. Eng. Inf. Technol, 2018. **8**(4-2): p. 1662-1671.
7. Yu, B., et al., *A survey of malware behavior description and analysis.* Frontiers of Information Technology & Electronic Engineering, 2018. **19**: p. 583-603.
8. Zhou, Y. and X. Jiang. *Dissecting android malware: Characterization and evolution*. in *2012 IEEE symposium on security and privacy*. 2012. IEEE.
9. Walenstein, A., et al. *Normalizing metamorphic malware using term rewriting*. in *2006 Sixth IEEE International Workshop on Source Code Analysis and Manipulation*. 2006. IEEE.
10. Andriatsimandefitra, R. and V.V.T. Tong. *Capturing android malware behaviour using system flow graph*. in *Network and System Security: 8th International Conference, NSS 2014, Xi'an, China, October 15-17, 2014, Proceedings 8*. 2014. Springer.
11. Enck, W., et al., *Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones.* ACM Transactions on Computer Systems (TOCS), 2014. **32**(2): p. 1-29.
12. Vidas, T., D. Votipka, and N. Christin. *All your droid are belong to us: A survey of current android attacks*. in *5th USENIX Workshop on Offensive Technologies (WOOT 11)*. 2011.
13. Lashkari, A.H., et al. *Toward developing a systematic approach to generate benchmark android malware datasets and classification*. in *2018 International Carnahan conference on security technology (ICCST)*. 2018. IEEE.

14. Bibi, I., et al. *An effective Android ransomware detection through multi-factor feature filtration and recurrent neural network*. in *2019 UK/China Emerging Technologies (UCET)*. 2019. IEEE.
15. Taheri, L., A.F.A. Kadir, and A.H. Lashkari. *Extensible android malware detection and family classification using network-flows and API-calls*. in *2019 International Carnahan Conference on Security Technology (ICCST)*. 2019. IEEE.
16. Noorbehbahani, F., F. Rasouli, and M. Saberi. *Analysis of machine learning techniques for ransomware detection*. in *2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*. 2019. IEEE.
17. Sandeep, H. *Static analysis of android malware detection using deep learning*. in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*. 2019. IEEE.
18. Ding, Y., et al. *A deep feature fusion method for Android malware detection*. in *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*. 2019. IEEE.
19. Imtiaz, S.I., et al., *DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network.* Future Generation computer systems, 2021. **115**: p. 844-856.
20. Haq, I.U., T.A. Khan, and A. Akhunzada, *A dynamic robust DL-based model for android malware detection.* IEEE Access, 2021. **9**: p. 74510-74521.
21. Zhu, H., et al., *SEDMDroid: An enhanced stacking ensemble framework for Android malware detection.* IEEE Transactions on Network Science and Engineering, 2020. **8**(2): p. 984-994.
22. Luo, W., et al. *Malicious HTTPS Traffic Classification Algorithm Based on DCGAN_1D-CNN*. in *2021 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*. 2021. IEEE.
23. Bayazit, E.C., O.K. Sahingoz, and B. Dogan. *Neural network based Android malware detection with different IP coding methods*. in *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. 2021. IEEE.
24. Kim, J., et al., *MAPAS: a practical deep learning-based android malware detection system.* International Journal of Information Security, 2022. **21**(4): p. 725-738.

25. Anıl, U., *Ağ trafiği analizi ile derin öğrenme tabanlı Android kötücül yazılım tespiti.* Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi, 2022. **37**(4): p. 1823-1838.
26. Fallah, S. and A.J. Bidgoly, *Android malware detection using network traffic based on sequential deep learning models.* Software: Practice and Experience, 2022. **52**(9): p. 1987-2004.
27. Yilmaz, A.B., Y.S. Taspinar, and M. Koklu, *Classification of Malicious Android Applications Using Naive Bayes and Support Vector Machine Algorithms.* International Journal of Intelligent Systems and Applications in Engineering, 2022. **10**(2): p. 269-274.
28. *Android Malware Dataset (CIC-AndMal2017)*. Available from: https://www.unb.ca/cic/datasets/andmal2017.html.
29. *Android Malware Dataset (CIC-InvesAndMal2019)*. Available from: https://www.unb.ca/cic/datasets/invesandmal2019.html.
30. *Google Play Store.* . Available from: https://play.google.com/store/apps.
31. *VirusShare*. Available from: https://virusshare.com/.
32. Mahindru, A. and A. Sangal. *Deepdroid: feature selection approach to detect android malware using deep learning*. in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. 2019. IEEE.
33. *The Drebin Dataset.*; Available from: https://www.sec.cs.tu-bs.de/~danarp/drebin/download.html.
34. *AndroZoo Dataset.*; Available from: https://androzoo.uni.lu/.
35. *DroidCollector Dataset*. Available from: https://loci.ujn.edu.cn/htdocs/DroidCollector/DroidCollector_Dataset.htm.
36. Wu, Q., X. Zhu, and B. Liu, *A survey of android malware static detection technology based on machine learning.* Mobile Information Systems, 2021. **2021**: p. 1-18.
37. Sanz, B., et al. *Puma: Permission usage to detect malware in android*. in *International joint conference CISIS'12-ICEUTE 12-SOCO 12 special sessions*. 2013. Springer.
38. Kim, H., et al., *Risk assessment of mobile applications based on machine learned malware dataset.* Multimedia Tools and Applications, 2018. **77**: p. 5027-5042.
39. Altaher, A., *Classification of android malware applications using feature selection and classification algorithms.* VAWKUM Transactions on Computer Sciences, 2016. **10**(1): p. 1-5.

40. *Android Malware Genome Project, 2014. http://www.malgenomeproject.org/*. Available from: http://www.malgenomeproject.org/.