Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Android Malware Detection Using Generative Adversarial Networks: A Deep Learning Approach

Inza naeem
Department of Computer Science, NFC Institute of Engineering and Technology, Multan, Pakistan

Ahmed naeem*
Department of Computer Science, NFC Institute of Engineering and Technology, Multan, Pakistan. Email: ahmad.naeem@nfciet.edu.pk

Nazia Khan
Multan Campus, Virtual University For vu reference-PMPP01, Multan, Pakistan

Naeem aslam
Department of Computer Science, NFC Institute of Engineering and Technology, Multan, Pakistan

Aliya Amin
Department of Computer Science, NFC Institute of Engineering and Technology, Multan, Pakistan

ABSTRACT

The Android operating system, the most widely used mobile platform in recent years, is a prime target for malware attacks. Conventional detection techniques have significant challenges due to the growing volume and complexity of Android malware, especially signature-based techniques that struggle with obfuscated code and zero-day threats. This article proposes a novel approach to Android malware detection that integrates Generative Adversarial Networks (GANs) with deep learning (DL) and machine learning (ML) approaches. The proposed methodology tackles two key problems: the lack of malware samples and the data discrepancy between malicious and benign datasets. By employing GANs for data augmentation, the work increases the number and diversity of training data, hence improving detection accuracy. Custom datasets were created by extracting static and dynamic properties from Android application packages (APKs) using tools like CuckooDroid. A Random Forest approach was employed to choose a total of 357 characteristics in order to ensure optimal model performance. The F1-score, accuracy, precision, and recall were used to evaluate the classification models that were trained on the GAN-generated data. Experimental results showed that the GAN-based model outperformed state-of-the-art

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189
http://amresearchreview.com/index.php/Journal/about

methods with a high accuracy of 99.10%. These findings imply that GANs can greatly enhance Android malware detection systems by enhancing the caliber and variety of datasets.

Keywords: Machine Learning, Generative Adversarial Networks, Malware Detection, deep learning

Introduction

Smartphones have changed over the last few years from being basic cell phones to advanced computers[1]. Compared to personal computers, they use less energy and are far more portable. This fact increases their use of the Internet for both home and business purposes, including Internet banking, online transactions, email, SMS, and MMS messages, and Internet browsing. The smartphone is a helpful tool in our daily lives because of all these features, but they also make it more susceptible to attacks by malicious apps, as shown in the Figure 1[2].

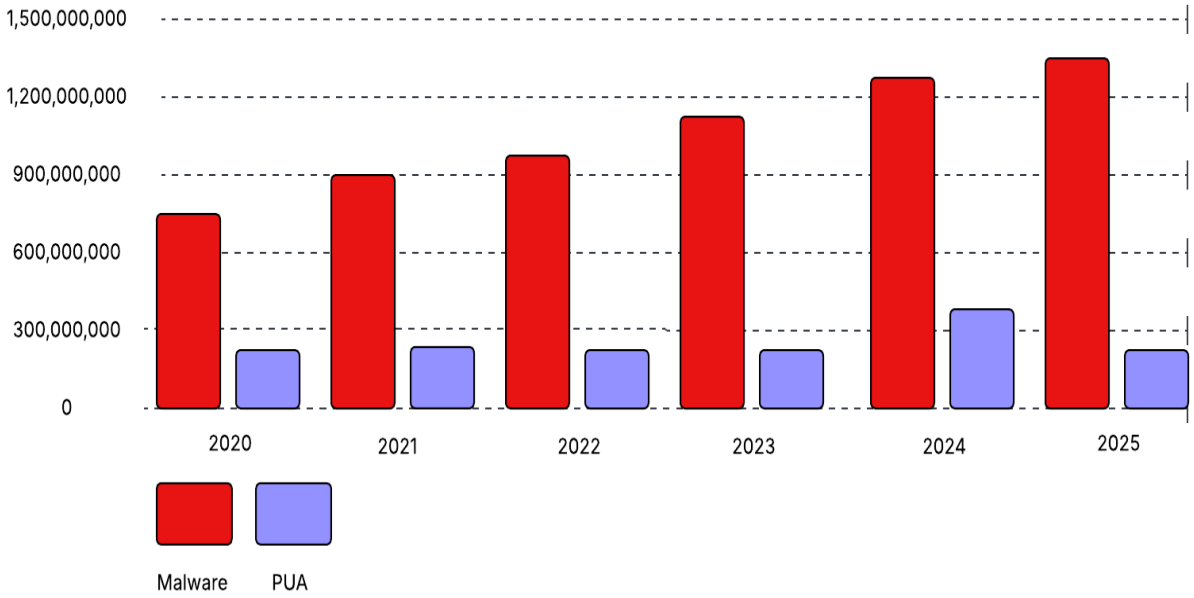

Figure 1: Total Number of Malware and PUA

Smartphones are a tremendously attractive target for attackers and malware producers since the majority of users save sensitive data on them, including phone numbers, emails, SMS messages, photos, and videos[3]. Android has become the most popular smartphone platform, with about 2.6 million apps available on Google Play alone. Because of the daily upgrades and the abundance of programs available, mobile application marketplaces like the Android Google Play have completely altered the way that users obtain software[4]. Both the sophistication and quantity of security threats targeting mobile platforms have increased as a result of the applications market's explosive growth and the ubiquitous nature of the apps offered on these platforms. Recent research indicates that dangerous or vulnerable programs are present in mobile

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

markets, endangering millions of devices. Malware has been a hazard to computer systems for a long time[5].

An estimated 1.2 billion Android users use the platform each month. With a 71.8% market share compared to Apple iOS's 27.6%, Android leads the industry[6]. This degree of expansion and acceptance of the Android platform is made possible by Google's rules. Millions of applications with a high degree of tolerance for verification and release are currently accessible on the platform due to its open policy[7]. The proliferation of Android malware can be attributed to a number of variables, some of which are not technological in nature. For example, free markets lack a regulating agency, and those who supply programs with vulnerabilities or malevolent potential face no repercussions. This situation is probably going to get worse as mobile applications become more common and sophisticated. Additionally, several of Google's rules pose a security risk to consumers[8].

The signature-based warning mechanism, for instance, is a popular detection method that matches known malware signatures to specific apps[9]. However, because a database could not have their signature, the technology is restricted in its ability to detect mobile malware, which is always evolving. Static, dynamic, hybrid, and machine learning malware detection techniques have become increasingly effective as a result of this conventional approach[10, 11].

This technique's scope includes figuring out whether or not a particular Android app is dangerous before it is released. Collaborative attacks from several apps operating together are not taken into account. Instead of concentrating on dynamic runtime behavior, the static analysis looks at the app file itself[12-14]. This implies that any compromised programs would go undetected after installation. In order to assist app store moderators, security firms, and analysts in promptly identifying harmful apps prior to their publication and user installation, this research attempts to create an effective method for Android malware detection[15, 16].

Malware is an undesirable and malevolent software program that targets mobile phones and damages them, in addition to causing private information to be lost or leaked[17]. The Symbian operating system was the victim of the first mobile virus, which was found in 2004. The first malware assault against Android was documented in 2010, and by 2011, new malware families were hitting every few weeks, making Android the most popular operating system for malware[18]. Malware for Android cellphones is pervasive and growing quickly.

Artificial Intelligence (AI) has been used in many different research domains since its inception and has emerged as a vital tool for virus identification. For example, methods like machine learning (ML) and deep learning (DL) are thought to be viable answers to malware detection issues like polymorphic attacks[19]. However, when using AI algorithms for mobile malware detection, two of the most significant problems arise from the need for high-quality training data. The first difficulty is gathering enough malware samples, particularly because methods like as DL need a huge number of samples. The second problem, on the other hand, is data imbalance in the obtained data, which might happen across benign and malignant samples or between malware families. In order to detect and categorize malware, we therefore suggest an Android malware detection framework that uses ML and DL models along with Generative

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189
http://amresearchreview.com/index.php/Journal/about

Adversarial Networks (GAN) for data augmentation. In addition to offering additional advantages like assisting in the discovery of zero-day malware, GANs are a useful way to deal with the problems of data imbalance and scarcity.

Literature Review

This part begins with a description of the three different machine learning approaches to detecting Android The authors K. Liu et al. [10] attempt to give a vivid and complete overview of research that employs machine learning-based algorithms to identify Android malware. Those begin with a review of the basics of Android apps, such as the system design, security, and classification of malware. Subsequently, they offer the publications surveyed and experiment with them through a multiplicity of criteria, starting with data pre-processing, then selecting the character features, and energy on machine learning algorithms. Last but not least, they provide burdensome detail concerning their difficulties and the recommendations they have of subsequent research.

In the meantime, W. Wang et al. [1] focus their attention on the features, which ML models utilize in the detection of malware on Android devices. They give an in-depth survey of existing study in the field and review of the 236 publications which have been selected along with the feature selection algorithms, features extracted, detection algorithm and evaluation measures taken. Then they give a taxonomy of the features and bring out the issues and the challenges encountered when utilizing them. They summarise their findings and recommendations of further work as they conclude their job.

DroidFusion [20] was produced by S. Y. Yerima and S. Sezer, and generates a classification model which can be employed in the detection of Android malware. They offer in their work a new method of classifier fusion based on multilevel architecture which enhances the performance of ML classes. The authors commence by researching pertinent literature and compare their proposed framework to five different fusion-based malware classifiers of Android malware. Then they enter the DroidFusion conjecture and discuss the experiments that they performed accordingly to check and evaluate the framework. After explaining and discussing the results of the experiment in detail, they remark that DroidFusion can only perform the binary classification and remind about the possibility of involving multiclass classification in their further research as one of the possible aims.

On the method level correlation relationship between the abstracted API calls of an application using that relationship, H. Zhang et al [21] offer a technique of detecting Android malware. Instead of call-graph based or frequency-based methods, they propose utilising the association rules of these abstracted API calls in order to record behavioral properties. The authors do not leave out detailed explanation on their proposed method and proceed to analyse and evaluate the results of their tests using the method. They indicate that in case the structure of the detection system is familiar, the proposed method can be attacked and is plagued with the pitfalls of static analysis. Then they give suggestions as to how the strictures may be met in later compositions.

MalResLSTM is a framework proposed by A. Alotaibi [22] that employs residual deep learning to identify and categorize malware by extracting complex features and hidden

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189
http://amresearchreview.com/index.php/Journal/about

structures from the Android package kit. After reviewing a wide range of relevant literature, the author goes into great detail to explain the theory underlying MalResLSTM. The Drebin dataset is used to test the model, and its results are evaluated and contrasted with those of competing ML and DL models. As a conclusion, author 5 highlights the attention approach, bidirectional LTSM, and the framework's vulnerability as possible areas for further research.

Semi-supervised learning is used by S. Mahdavifar et al. [23] to lower the costs of Android malware detection because training models requires huge amounts of tagged malware data, which can be expensive and challenging to collect. They present CICMalDroid2020, a new dataset of Android applications that include both static and dynamic attributes. They next train their model on the dataset using CopperDroid for pseudo-labelling and dynamic analysis. After that, they assess the performance of their model and contrast it with label propagation. They offer some recommendations for future research, including testing the recommended system on deep learning models, IoT devices, and other characteristics.

In order to deal with limited data for Android malware detection, Y. Chen et al. [6] look into the advantages of employing data augmentation techniques, specifically GAN and simple image augmentation methods. They examine the methods of their suggested framework and discuss a number of comparable publications. They then thoroughly explain their experiment and its findings before assessing and condensing them. They observe that despite preserving the original features of the dataset, data augmentation aids in enhancing the detection model's effectiveness, particularly for uncommon malware variants. Additionally, they discovered that the GAN's improvement was more noticeable than that of methods like image flipping and rotation, and they hypothesized that this is because the generated samples were different. They mention possible future study directions and acknowledge the limits of their findings in their paper's conclusion.

J. Li et al. introduce SynDroid: a SynDroid is model to classify malware in the Android system using CTGAN-SVM [19]. They recommend the use of CTGAN to provide quality samples and SVM to reject poor samples. Also, they provide the KS-CIR, the test which evaluates the samples in terms of quantity and quality with the aim of finding out which categories are going to be improved. Authors conduct experiments with the CCCS-CIC-AndMal2020 Dataset and Random Forest (RF) model. Measurable points such as accuracy, recall, precision, and F1-score are taken to contrast the outcomes of the model concerning the performance of the model pre- and post-augmentation. The results illustrate that their strategy enhances performance and that it decreases the effect of data imbalance.

Table 1: Related Work

Author	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Raza, Ali, et al.[29]	TL-GNN	98.87	99.55	97.30	99.42
Smmarwar et al.[30]	OEL-AMD	96.95	95.99	94.89	95.98

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189
http://amresearchreview.com/index.php/Journal/about

Go et al. [3]	ResneXt	98.32	97.64	97.93	97.69

Methodology

This chapter describes the approach taken in the study to use Generative Adversarial Networks (GAN) for Android malware detection. It describes how to prepare datasets, extract features, and apply the GAN model to data augmentation. This chapter aims to provide a concise and organized outline of the actions taken to achieve the study's goals and address the suggested questions.

5831 malicious and 5546 benign Android apps were among the data sets used in the studies. Table 2 displays the format of the benign apps, which are downloaded from AndroZoo [54], a large database of Android apps that offers a comma-separated values (CSV) file with information on every app. The malware samples originate from Drebin [55], which includes 179 families and 5560 Android malware variants. 5546 harmful samples were successfully converted to RGB images in this work; Table 3 displays the family number statistics.

The malware data set is separated by family, and the test and training sets (including the validation set) have an 8:2 ratio. The classification experiment is conducted on the No. 1 through No. 12 data sets.

Feature Processing

As code obfuscation does not change the system methods, system API sequences represent a good illustration of the Android application behavior. That is why such features as system methods are so valuable, being able to reflect the real meaning of an application. Besides the system calls (Android platform APIs), Android apps however use both external interfaces (e.g. to third-party services) and internal techniques specified by the developers. Malicious code often does not allow the uniform and unique representation of features as inconsistencies employed in naming conventions and other techniques used to obfuscate a code can undermine this. To accommodate this, we will take the use of system API sequences, which will depict the behavior that we want the applications to do. We substitute internal procedures and external calling interfaces with their corresponding result calls (system API sequences), in sorted order; by the depth-first search (DSP) algorithm of traversal. The method guarantees the avoidance of memory of the behavior logic and the constant depiction of application activity. Lastly, the resulting API sequences are converted into RGB images through a look-up in a pre-constructed feature database of color-coded feature values. Figure 2 shows how this visualization process can be performed.

We utilize Check Point Software Technologies' CuckooDroid to automate the feature extraction procedure. Based on the well-known Cuckoo sandbox and a number of other open-source projects, CuckooDroid is an automated, cross-platform framework for emulation and analysis. It offers both static and dynamic APK inspection, avoidance of specific VM-detection methods, encryption key extraction, SSL inspection, API call trace, basic behavioral signatures, and many other features. The procedure for examining each APK sample and extracting features is depicted in Fig. 3 below.

After the APK files are uploaded to CuckooDroid, the analysis, feature extraction process shown in Figure 3 starts. Through the Android Debug Bridge (ADB), each APK

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

is launched on the emulator, and the live execution is tracked in the Android Studio based emulator interface. After analysis, automatic generation of JSON report occurs, which provides a result of the static as well as dynamic analysis. Equipped with processed APK files, they will be classified into two directories; into one with benign applications and the other having malicious sample. The resulting JSON files generated by the benign apps are placed in the corresponding folder and after which malware APK analysis is done and the respective JSON reports are kept accordingly as well. To qualify the success of the analysis, the execution logs closely examined to validate that the each APK has been successfully installed and executed in the emulator before its JSON report is stored as shown in Figure 2.

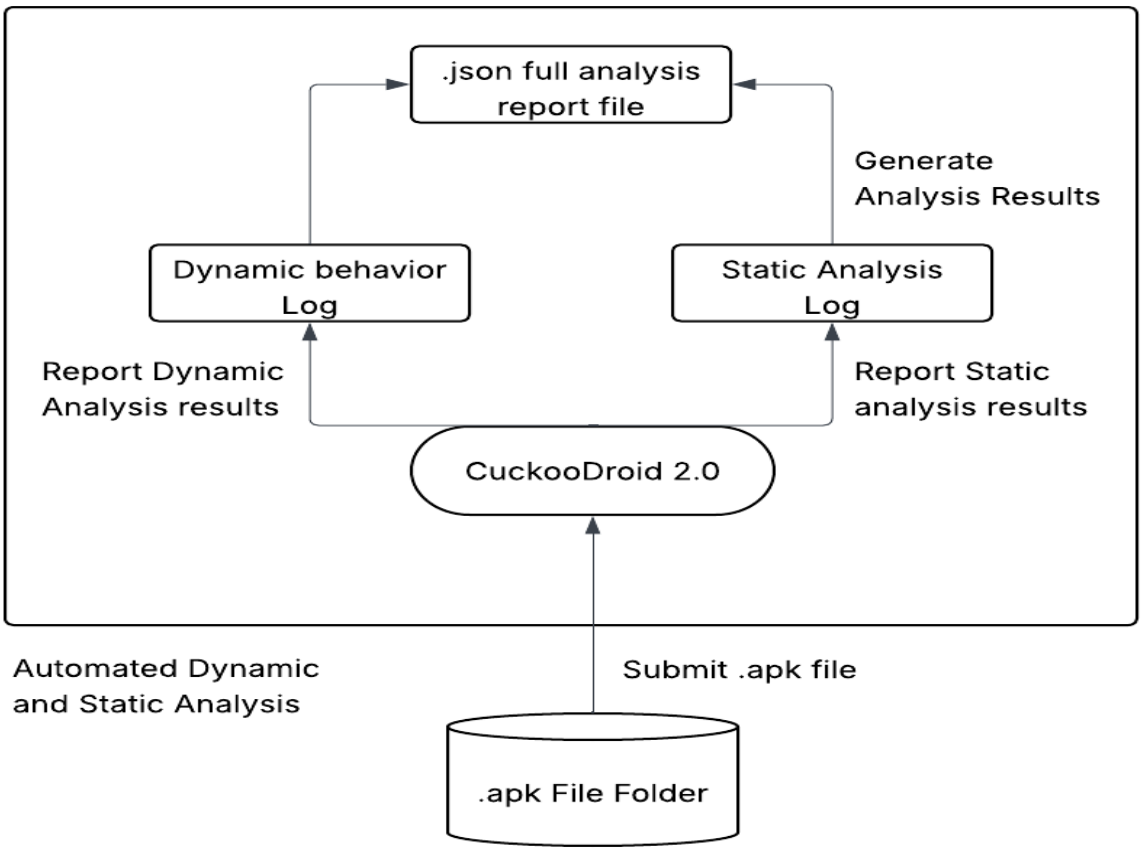

Figure 2: APK sample and extracting features

In the process of developing a useful and trustworthy machine learning model, one of the significant stages would be the selection of the critical features that would have a significant impact on the model. Many features tend to raise the computation expense and processing time but not necessarily enhance the performance. In this paper, feature selection technique narrowed the 714 features (achieved during static and dynamic analysis) to 357 most relevant characteristics used to gain the capability to classify correctly the biggest number of objects. The goals of feature selection were 3-fold: to

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189
http://amresearchreview.com/index.php/Journal/about

improve the predictive performance of the classifiers, to enhance efficiency by decreasing the computational overheads, and to explore in more detail the underlying processes that are generating those data. In order to do this, we utilized the Random Forest (RF) methodology which attaches an importance score to features relative to their prediction contribution to classification. A Random Forest feature selection is of the type of embedded methods, or a hybrid between a filter and a wrapper method, where feature selection is built directly into the learning algorithm. This method has a number of advantages such as the enhancement of generalization, enhanced interpretability and high accuracy. In the process we have come up with 357 important attributes, the 20 features with the highest key attribute scores, representing the analytical nucleus.

Proposed Approach

Among the efforts to make conventional machine learning-based malware classifications perform better, this study introduces an Android malware classification framework built on the functionality of a Generative Adversarial Network (GAN). The basic idea is to learn artificial feature vectors of GANs that are alike malware. This complements the training set and will also contribute to training a more effective and general detecting mechanism. Two networks that are vital to the basic setup of a GAN are the discriminator model, and the generator model. In GANs, adversarial means that both of these networks are trained concurrently and are somewhat competitive, as in a zero-sum game such as chess. The overarching idea behind the generator is to generate new images which are realistic to unexpectedly fool the discriminator. In the simplest GAN setup to produce pictures, the generator takes a message of random noise and produces an image message as output.

The discriminator, as the "opponent" to the creative efforts of the generator, on the other hand, is related to the task of a binary image classifier, which has the responsibility to distinguish between genuine and fraudulent pictures. Due to this constant back-and-forth interaction method, the generator will eventually become better at generating natural-looking images, whereas the discriminator will become better at differentiating between actual and product-generated images. One of the main advantages of GANs is that they allow generating high-quality synthetic data. Due to such collaboration of the discriminator and generator, the generator has the possibility to learn the feedback of the discriminator and come up with several data points that cannot be distinguished much with the actual thing. Moreover, in comparison with more traditional methods, GANs tend to be rather fast and efficient. The reason is that, they use parallel neural networks in computation due to opportunities of parallelization that it offers.

To conclude, Simple GAN architecture has three main components, the discriminator that ecstatically evaluates the authenticity of real and unreal pictures, the generator, which creates unreal pictures, and the genuine pictures in training set. GANs use a strategy that contrasts with that of most deep learning models, which can be trained to minimize a single cost (in image classification). The discriminator and the generator in GANs have contrasting objectives, as well as cost functions. The aim of the discriminator is to successfully distinguish between real and fake ones, whereas that of the generator is to create fake images that would closely resemble real ones so that they

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189
http://amresearchreview.com/index.php/Journal/about

could deceive the discriminator. Unlike ordinary optimization-based models, GANs are in an adversarial relationship.

Both the discriminator and the generator improve their respective capacities as training goes on. The discriminator becomes better at correctly differentiating between actual and false images, while the generator gets better at creating images that closely mimic the training data. The GAN training dynamic is characterized by this process of constant development.

Achieving a fine balance in the game is important to GAN training. The goal is for the discriminator to eventually lose its ability to discriminate between real and false images, while the generator produces data that is remarkably similar to the training data. This equilibrium shows that the GAN model was successfully trained, as shown in the Figure 3.

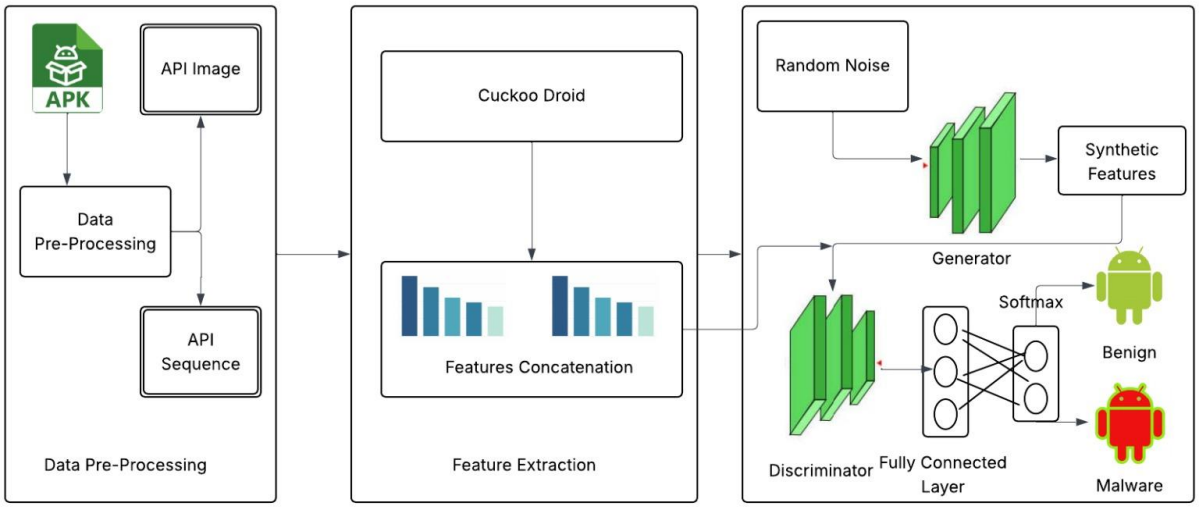

Figure 3: Working Flow

Results and Discussion

This section compares the results of our suggested approach with the state-of-the-art and details the experimental setup and evaluation metrics.

The research was conducted using the Jupyter Notebook development environment. The models for the dataset pre-processing were created and trained using Google Collaboratory, or simply "Colab." Additionally, the Integrated Development Environment (IDE) Anaconda was utilized. The goal of this version of R and Python, two scientific computer programming languages, is to simplify package administration and deployment.

- A system with the following requirements for pre-processing 64-bit operating system, specifically running Windows 11, and using an Intel(R) Core (TM) i5 CPU.
- A 1TB solid-state disk drive and 16 GB of RAM

Comparison with the State-of-the-Art Model

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

Our suggested GAN pre-training model demonstrated high evaluation metrics on datasets, according to the experimental results. The GAN model demonstrated comparable performance on the dataset, attaining evaluation metrics exceeding 99.10%, as indicated in Table 3.

Table 2: Comparison table with state-of-the-art approaches

Author	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Raza, Ali, et al.[29]	TL-GNN	98.87	99.55	97.30	99.42
Smmarwar et al.[30]	OEL-AMD	96.95	95.99	94.89	95.98
Go et al. [3]	ResneXt	98.32	97.64	97.93	97.69
Proposed Approach	GAN	99.10	99.15	99.07	99.11

The comparison of GAN and TL-GNN is displayed in the figure. Using assessment measures to assess our model's success, we contrasted our model with our recommended methodology. The results of TL-GNN are inferior to our suggested method in terms of accuracy, precision, recall, and f-measure. With an accuracy of 98.87%, a precision of 99.55%, a recall of 97.30%, and an F-measure of 99.42%, the GAN model fared better than the TL-GNN model. With an accuracy of 99.10%, precision of 99.15%, recall of 99.07%, and F-measure of 99.11%, the GAN model outperformed the others, as shown in Figure 4.

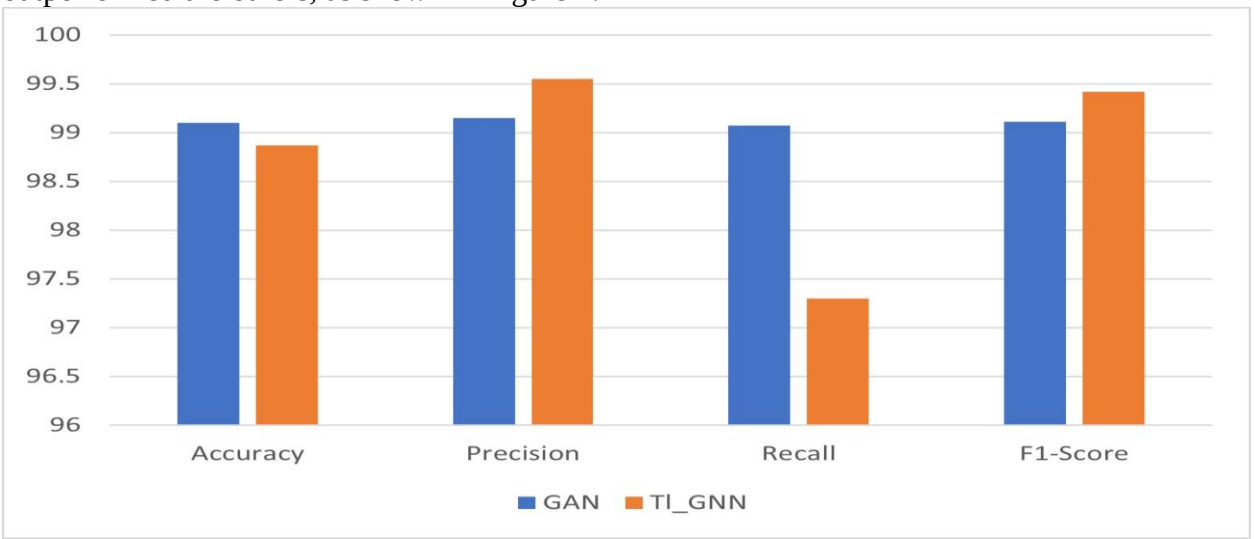

Figure 4: Comparison with TL_GNN

The comparison between OELAMD and GAN is displayed in the figure. Using assessment measures to assess our model's success, we contrasted our model with our recommended methodology. The outcomes of OEL-AMD are inferior to our suggested method in terms of accuracy, precision, recall, and f-measure. With accuracy of 96.95%, precision of 95.99%, recall of 94.89%, and an F-measure of 95.98%, the GAN

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189
http://amresearchreview.com/index.php/Journal/about

model fared better than the OEL-AMD model. With an accuracy of 99.10%, precision of 99.15%, recall of 99.07%, and F-measure of 99.11%, the GAN model outperformed the others, as shown in Figure 5.

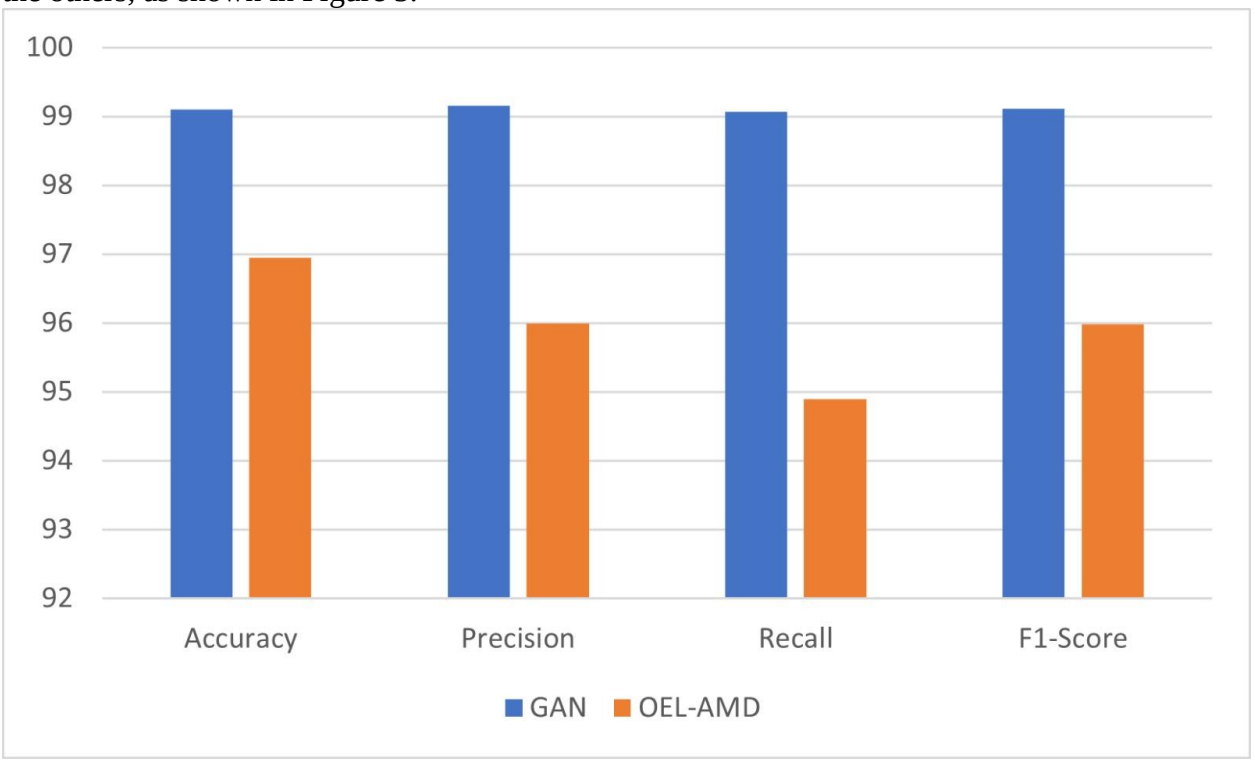

Figure 5: Comparison with OEL-AMD

The GAN and ResneXt comparison are displayed in the figure. Using assessment measures to assess our model's success, we contrasted our model with our recommended methodology. The results of ResneXt are inferior to our suggested method in terms of accuracy, precision, recall, and f-measure. With an accuracy of 98.32%, precision of 97.64%, recall of 97.93%, and F-measure of 97.79%, the GAN model performed better than the ResneXt model. With an accuracy of 99.10%, precision of 99.15%, recall of 99.07%, and F-measure of 99.11%, the GAN model outperformed the others.

ROC curve for a 99.10% accurate GAN discriminator. The AUC (Area Under Curve) is close to 1, suggesting great classification performance, and the curve hugs the top-left corner, indicating a high true positive rate and a low false positive rate. Please let me know if you would like a comparison with another model or the PR curve as well, as shown in Figure 6 and Figure 7.

Annual Methodological Archive Research Review
http://amresearchreview.com/index.php/Journal/about
Volume 3, Issue 8 (2025)

Online ISSN 3007-3197
Print ISSN 3007-3189

http://amresearchreview.com/index.php/Journal/about

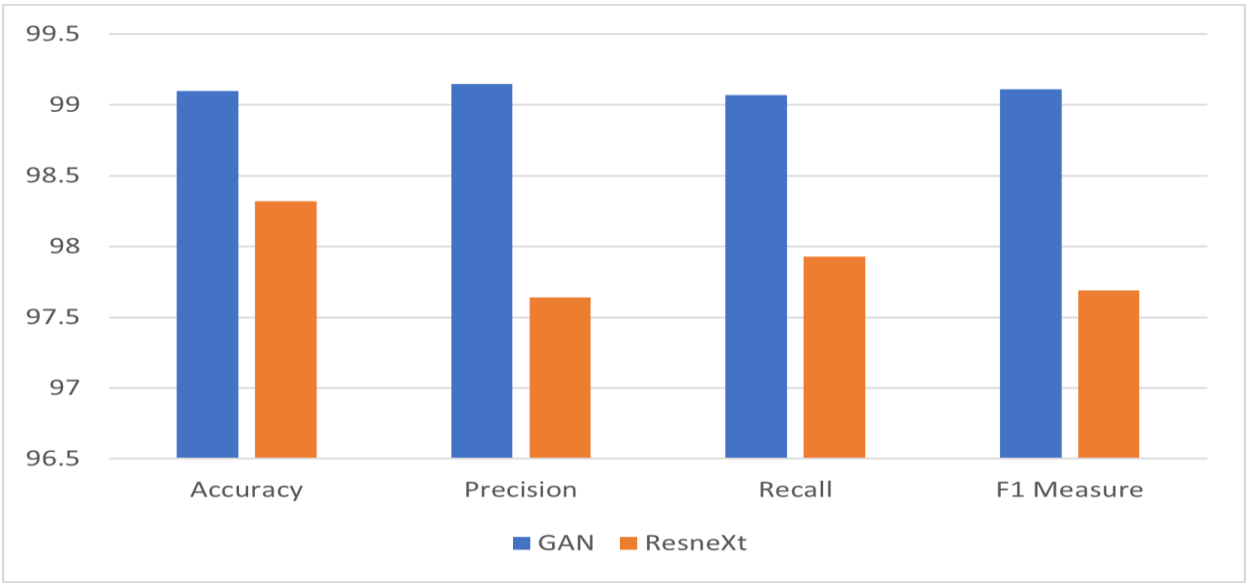

Figure 6: Comparison with ResNeXt

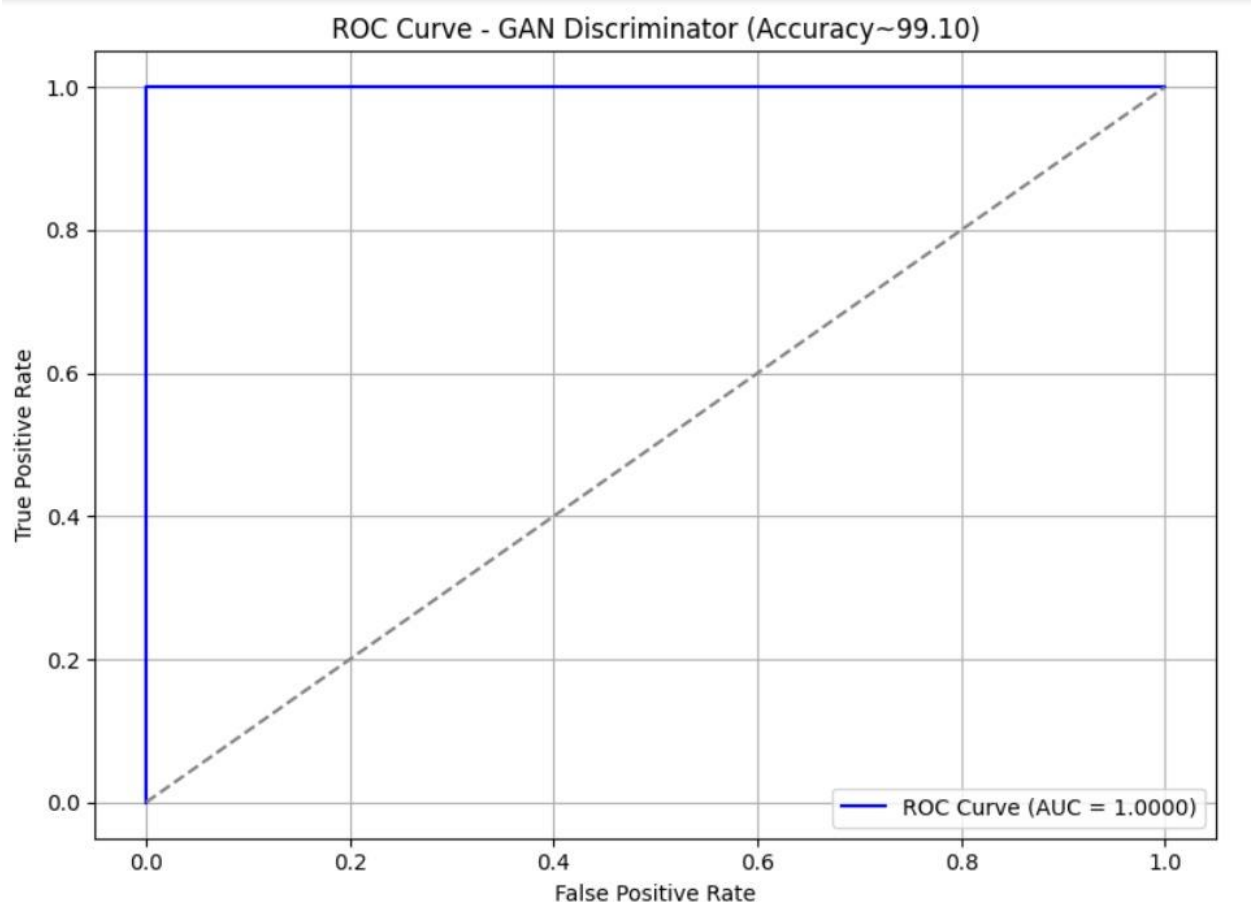

Figure 7: ROC Curve

Conclusion and Future Work

Online ISSN 3007-3197
Print ISSN 3007-3189

A GAN-based framework for identifying Android malware was effectively created and assessed in this study. The results demonstrated that the performance of conventional ML and DL models is much improved when GAN-generated synthetic data is combined with static and dynamic feature analysis. The application of GANs enhanced the model's capacity to identify complex threats including Trojan horses, botnets, adware, and malicious VPN programs while addressing common problems in malware detection, such as data imbalance and scarcity. Accuracy, precision, recall, and F1-score were among the evaluation measures that consistently showed that the suggested GAN model provides a higher detection rate than other models, such as TL-GNN, OEL-AMD, and Bi-LSTM+CNN. Furthermore, the system was able to specialize in identifying subtle dangerous behaviors that generic datasets can miss by employing specialized datasets that were focused on particular malware types. All things considered, the study provides a scalable and efficient method for early Android malware detection, which can be very helpful for cybersecurity companies and app store moderators.

There are numerous chances to improve and broaden this research, notwithstanding the remarkable results of the suggested GAN-based architecture for Android malware detection. The incorporation of real-time analysis is a crucial topic for further research. Because the model currently uses offline dynamic and static data, it might not be able to identify malware that only activates during runtime or after installation. The ability of the system to identify such covert threats can be greatly enhanced by integrating real-time behavioral monitoring. The detection of collaborative malware attacks, in which several programs cooperate to carry out destructive actions, is another crucial avenue. Since the current model assesses each application separately, it might be possible to further enhance security by extending it to find patterns of joint behavior and inter-app communication. Additionally, the framework may be modified for Android-based smart devices and the Internet of Things (IoT), providing safety for wearables and smart homes in addition to mobile phones.

Furthermore, the implementation of federated learning may enable decentralized model training across several devices while protecting user privacy, which is becoming a more crucial factor in cybersecurity. In order to make the malware detection process more transparent and understandable for analysts and security experts, future research can potentially investigate the integration of explainable AI (XAI) techniques. Last but not least, broadening the dataset to encompass more malware families—particularly more recent and uncommon varieties—would enhance the model's resilience and generalizability even more. These improvements have the potential to provide the current architecture with a complete and more flexible defense against changing Android malware threats.

References:

Wang, Z., Q. Liu, and Y. Chi, Review of android malware detection based on deep learning. Ieee Access, 2020. **8**: p. 181102-181126.

Amin, M., et al., Android malware detection through generative adversarial networks. Transactions on Emerging Telecommunications Technologies, 2022. **33**(2): p. e3675.

Online ISSN 3007-3197
Print ISSN 3007-3189
http://amresearchreview.com/index.php/Journal/about

Go, J.H., et al. Visualization approach for malware classification with ResNeXt. in 2020 IEEE Congress on Evolutionary Computation (CEC). 2020. IEEE.
Alshebli, S.A., GAN-based Enhancer for Malware Detection on Android Platform. 2024, Khalifa University of Science.
Alshebli, S., et al., Enhanced Android Malware Detect Models based on Explainable Generative Adversarial Networks. IEEE Access, 2025.
Thu, L.B., et al., A survey on generative adversarial networks for malware analysis. Journal of Computer Science and Cybernetics, 2025.
Islam, R., et al., Android malware classification using optimum feature selection and ensemble machine learning. Internet of Things and Cyber-Physical Systems, 2023. **3**: p. 100-111.
Liu, J., et al., Adversarial machine learning: A multilayer review of the state-of-the-art and challenges for wireless and mobile systems. IEEE Communications Surveys & Tutorials, 2021. **24**(1): p. 123-159.
Deldar, F. and M. Abadi, Deep learning for zero-day malware detection and classification: A survey. ACM Computing Surveys, 2023. **56**(2): p. 1-37.
Liu, Y., et al., Deep learning for android malware defenses: a systematic literature review. ACM Computing Surveys, 2022. **55**(8): p. 1-36.
Qiu, J., et al., A survey of android malware detection with deep neural models. ACM Computing Surveys (CSUR), 2020. **53**(6): p. 1-36.
Mazaed Alotaibi, F. and Fawad, A multifaceted deep generative adversarial networks model for mobile malware detection. Applied Sciences, 2022. **12**(19): p. 9403.
Baghirov, E., Building robust malware detection through conditional Generative Adversarial Network-based data augmentation. Программные системы: теория и приложения, 2024. **15**(4): p. 97-110.
Dhull, M. and C. Rana, An Empirical Study on the Effectiveness of Adversarial Examples in Window PE Malware Detection Model. 2025.
Kc, B., S. Sapkota, and A. Adhikari, Generative adversarial networks in anomaly detection and malware detection: a comprehensive survey. Advances in Artificial Intelligence Research, 2024. **4**(1): p. 18-35.
Nazir, A., et al., Collaborative threat intelligence: Enhancing IoT security through blockchain and machine learning integration. Journal of King Saud University-Computer and Information Sciences, 2024. **36**(2): p. 101939.
Yan, H., et al., Malware Detection Method Based on Feature Fusion. 2024.
Yan, H., et al., Malware classification method based on feature fusion. International Journal of Information Security, 2025. **24**(2): p. 1-17.
Li, J., et al., SynDroid: An adaptive enhanced Android malware classification method based on CTGAN-SVM. Computers & Security, 2024. **137**: p. 103604.
Yerima, S.Y. and S. Sezer, Droidfusion: A novel multilevel classifier fusion approach for android malware detection. IEEE transactions on cybernetics, 2018. **49**(2): p. 453-466.
Zhang, H., et al., An efficient Android malware detection system based on method-level behavioral semantic analysis. IEEE Access, 2019. **7**: p. 69246-69256.
Alotaibi, A., Identifying malicious software using deep residual long-short term memory. Ieee Access, 2019. **7**: p. 163128-163137.

Online ISSN 3007-3197
Print ISSN 3007-3189

Mahdavifar, S., D. Alhadidi, and A.A. Ghorbani, Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder. Journal of network and systems management, 2022. **30**(1): p. 22.

Burks, R., et al. Data augmentation with generative models for improved malware detection: A comparative study. in 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON). 2019. IEEE.

Bae, J. and C. Lee. Easy data augmentation for improved malware detection: A comparative study. in 2021 IEEE International Conference on Big Data and Smart Computing (BigComp). 2021. IEEE.

Phung, N.M. and M. Mimura. Data augmentation of JavaScript dataset using DCGAN and random seed. in 2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW). 2021. IEEE.

Won, D.-O., Y.-N. Jang, and S.-W. Lee, PlausMal-GAN: Plausible malware training based on generative adversarial networks for analogous zero-day malware detection. IEEE Transactions on Emerging Topics in Computing, 2022. **11**(1): p. 82-94.

Stalin, K. and M.B. Mekoya, Improving android malware detection through data augmentation using wasserstein generative adversarial networks. arXiv preprint arXiv:2403.00890, 2024.

Raza, A., et al., TL-GNN: Android Malware Detection Using Transfer Learning. Applied AI Letters, 2024. **5**(3): p. e94.

Smmarwar, S.K., et al., An optimized and efficient android malware detection framework for future sustainable computing. Sustainable Energy Technologies and Assessments, 2022. **54**: p. 102852.