Sistemasi: Jurnal Sistem Informasi ISSN:2302-8149
Volume 15, Nomor 5, 2026: 1777-1795 e-ISSN:2540-9719

GAN-CNN-based Android Ransomware Detection System using Network Traffic Analysis

[1]Mahmood S. Mahmood*
[1]University of Mosul, College of science, Department of Forensic evidence
Mosul - Iraq
*e-mail: mahmoodsubhy1981@uomosul.edu.iq

(***received*:** 26 April 2026, ***revised*:** 16 May 2026, ***accepted*:** 17 May 2026)

Abstract

Android ransomware poses a major threat to cybersecurity, resulting in financial losses, data thefts, and service disruptions for mobile users. In this paper, a network traffic-based ransomware detection framework is proposed, which combines the feature selection and data augmentation approaches with machine learning and deep learning algorithms. The proposed methodology consists of data preprocessing, data normalization, class balancing, and feature reduction based on the Random Forest importance and SHAP analysis to select the most informative features. Different classification models such as Logistic Regression (LR), Decision Tree (DT), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), TabNet, Deep Neural Network (DNN), and Convolutional Neural Network (CNN) are evaluated and compared. Generative Adversarial Networks (GANs) are used to generate synthetic ransomware samples for training, to cope with class imbalance, and to enhance detection capability. The results of the experiments proved that the GAN-improved CNN model's overall accuracy is 99.5%, recall is 99.8%, precision is 99.6%, F1 score is 99.6%, and AUC is 98.9%. The results further show that feature reduction resulted in reduced time in training and testing with high detection performance. This paper emphasizes the importance of the proposed feature selection, augmentation using GAN, and deep learning approach for detecting Android ransomware. The framework proposed, however, led to decreased feature space and increased computational efficiency, but additional testing on real Android devices is still needed to confirm the claims of lightweight deployment and low resource usage.

Keywords: android ransomware, CNN, deep learning, GAN, intrusion detection system, machine learning, network traffic analysis

1 Introduction

Smart devices have reached a stage where they are one of the most significant things in contemporary life due to offering an exceptional amount of new capabilities that enable users to get high-quality personalized services on their mobile platforms [1]. These services will be voice and video services, email, web browsing, online shopping, and mobile banking. Many operating systems are compatible with smart device functionality, the most common of them being Android [2]. Security concerns about the Android operating system have also been on the increase as its use continues to grow. Where the percentage of data that is stored and transferred via the Android operating system is large and most of it is personal and sensitive. In addition, malware developers and cyber attackers usually use the vulnerabilities of the Android platform to achieve unauthorized access to this data. Moreover, the open-source lifestyle of Android, although promoting innovativeness, also leaves the system vulnerable, as it is prone to malicious attacks [3]. Furthermore, a large number of applications are developed for the Android platform, some of which have vulnerabilities that are exploited by attackers to carry out malicious activities. Some of these activities include ransomware attacks, the installation of backdoors, phishing attempts, unauthorized sending of premium SMS messages, and the theft of personal data [4].

Ransomware is one of the most common threats that attack mobile devices. In this type of attack, attackers either encrypt the information on the machine or lock out the machine, which in effect blocks legitimate users of their data or system functionality until they can pay some ransom [5]. The impact of such attacks may be of significant financial and operational impacts, often leading to

 ISSN:2302-8149
e-ISSN:2540-9719

extensive data breaches, long system outages, and reputational losses to organizations [6]. Mobile ransomware represented a notable threat in 2024. During the second quarter alone, Kaspersky identified 1,392 malicious installation packages associated with mobile ransomware Trojans. A prominent example was the Rasket Trojan, which leveraged Tasker automation scripts and was particularly active in early 2024, although its activity declined by the end of the first quarter.

Various layers of security mechanisms, such as firewalls, antivirus software, and Intrusion Detection Systems (IDSs), are employed to safeguard against cyberattacks [7]. Firewalls are primarily designed to regulate traffic between networks; however, they do not generate alerts in the event of an internal attack [8]. Antivirus software is generally effective in identifying known malware but performs poorly against previously unseen threats. Intrusion Detection Systems (IDSs) are typically classified into two categories: signature-based and anomaly-based detection. Signature-based IDSs detect malicious behavior by comparing it to pre-defined attack signatures stored in a database. However, their effectiveness is limited when identifying novel threats, as they rely on frequent updates to remain current. Anomaly-based IDSs, on the other hand, establish a baseline of normal system behavior and identify deviations from this model as potential threats. This approach is more adept at identifying new or unknown malware attacks [9]. The primary objective of an Intrusion Detection System (IDS) is to identify a wide range of malicious software (including botnets, Trojan horses, spyware, backdoors, worms, ransomware, and riskware) as fast as possible. This is the kind of detection that cannot be performed by the traditional firewalls alone [10].

Anomaly-based intrusion detection systems (IDSs) in the Android operating system analyze and interpret malware activity using three main methods: static, dynamic, and hybrid analysis. In static analysis, an Android application package is decompiled to extract important files (particularly classes.dex and AndroidManifest.xml) from which different features are retrieved without running the code. According to Malik, these features could include opcodes, permissions, intents, API calls, and utilized components. Dynamic analysis, on the other hand, collects runtime information regarding system calls, network traffic, and hardware resource usage—such as CPU, memory, and battery consumption—while the application is running to identify malicious activity. The hybrid technique merges the advantages of both static and dynamic approaches [11,12].

Android malware detection models are typically built by machine learning (ML) methods [13]. Machine learning (ML) techniques can be used to trace the network traffic to detect the ransomware operations, which is one of the most effective methods to do it [14]. Analysis of such traffic can reveal the behavioral patterns of ransomware, such as abnormal and unstable file transfers and abnormal and unstable communications with command and control (C2) servers [6].

Although the number of research studies related to Android ransomware detection is increasing, existing approaches still have some limitations. A large number of existing studies employ high-dimensional sets of features and models, which can require significant amounts of computation and may be of limited use on mobile devices with limited resources. Furthermore, class imbalance and the scarcity of samples with a proper representation of ransomware can impact the reliability and generalization ability of a model in detecting new or emerging ransomware variants. Therefore, this paper develops an accurate and computationally efficient Android ransomware detection system based on network traffic analysis while minimizing the impact of high-dimensional data and class imbalance. To address this problem, this paper proposes a framework that integrates data preprocessing, feature selection using Random Forest and SHAP, GAN-based data augmentation, and comparative evaluation of several machine learning and deep learning algorithms. The main objectives of this paper are: (1) to identify the most relevant network traffic features for ransomware detection, (2) to improve classification performance by addressing class imbalance using GAN-generated samples, and (3) to evaluate the effectiveness of multiple classification models in detecting Android ransomware based on network traffic data.

The main contributions of this paper are as follows:

- The preprocessing stage, feature selection, data augmentation, and classification stages are combined into a single workflow, resulting in a unified Android ransomware detection framework based on network traffic analysis.
- To achieve a high detection rate with a reduced number of features in the network traffic, feature reduction has been applied using random forest and SHAP. Random Forest and SHAP have been

 ISSN:2302-8149
e-ISSN:2540-9719

employed for feature reduction to achieve high detection performance with a reduced number of features in network traffic.

- To address the issue of class imbalance in the ransomware dataset and to enhance the training of deep learning algorithms, GAN-based augmentation is used. GAN-based augmentation is used to address the issue of class imbalance in the ransomware dataset and to enhance the process of training deep learning algorithms.
- Several ML and DL algorithms (LR, DT, KNN, SVM, MLP, TabNet, DNN, and CNN) are tested and compared with the same experimental setup.
- The results of the study showed that the overall performance of the GAN + CNN model has been the best on the data set used and has been tested with high accuracy and an improved recall rate compared to other models used in the evaluation.

The remainder of this paper is organized as follows: Section 2 provides the background, covering the concept of ransomware, machine learning algorithms, feature selection methods, and ransomware datasets. Section 3 reviews related work, while Section 4 presents a results discussion of the proposed model.

2. Background

To create a successful ransomware detector, it is important to have a clear grasp of several essential concepts and the dataset. The subsections below expound on the ransomware concept, machine learning algorithms, feature selection methods, and the dataset used in the training and testing of a powerful Android ransomware detection model.

2.1. Ransomware Concept

Ransomware is a major cybersecurity risk that affects individuals, corporations, organizations, and even the government. It operates on the principle of locking or encrypting data, systems, or devices and then requires a ransom to be paid back to unlock access. Recently, ransomware attacks have become more deadly and expensive. WannaCry attacked in 2017 and caused damages worth more than $300 million. A ransomware attack on a German hospital in 2020 caused the first known death as a result of a ransomware attack on a health facility [15,16]. By 2031, experts estimate ransomware damages will add up to $265 billion per year [17].

Ransomware is mainly classified into crypto-ransomware and locker-ransomware. A crypto-ransomware locks the victim's files and asks for a ransom to unlock them, and a locker-ransomware locks the entire device. Typical file types that are targeted by crypto-ransomware are important documents, images, and PDFs with sensitive information. There are various types of ransomware detection methods, such as statistical, event-driven, data-centric, and machine learning-based detection techniques. In this study, the machine learning approach is used to detect Android ransomware by analyzing network traffic.

2.2. Machine Learning Algorithms

Machine learning is a computational approach that enables software systems to improve predictive accuracy by refining algorithms based on data, without the need for explicit programming instructions. These algorithms primarily function by generating outputs through predictive modeling of input data, rather than relying on predefined, static procedures. The models are continuously updated based on the output to enhance performance [11]. Commonly used machine learning algorithms include the Random Forest (RF) algorithm [18], Logistic Regression (LR) [19], Decision Tree (DT) [20], K-Nearest Neighbors (KNN) [21], Multilayer Perceptron (MLP) [22], Support Vector Machines (SVMs) [23], Tabular Networks (TabNet) [24], Convolutional Neural Networks (CNNs) [25], and Generative Adversarial Networks (GANs), which generate new data (as real data) that approximates existing data with the help of two neural networks [26].

2.3. Datasets

Although machine learning and deep learning algorithms have significant potential in identifying ransomware malware, they are faced with a number of problems and shortcomings when selecting the dataset for training and testing. The small number of available malware samples is a major problem since most of them are scarce by nature, and their malicious nature causes a lack of labeled datasets

 ISSN:2302-8149
e-ISSN:2540-9719

[27]. Also, there are challenges with obtaining and analyzing ransomware samples. These are due to advanced encryption and obfuscation techniques of ransomware that make it difficult to detect and classify [28].

The CICAndMal2017 dataset is a standard Android malware dataset designed by the Canadian Institute for Cybersecurity to aid in machine learning-based detection of malware. It is composed of more than 10,000 applications, both benign and malicious (1,048,574 benign samples and 460,976 ransomware samples), of which there are over 80 flow-based features gathered in the real world. The categories into which the malware samples are classified include adware, ransomware, scareware, and SMS malware, with dozens of malware families. These drawbacks are class imbalance issues, problems of feature redundancy/correlation, and performance sensitivity [29].

The CICAndMal2019 dataset is a refined form of the previous Android malware datasets, and it offers both dynamic and static analysis capabilities to enhance malware detection. It has information on API calls, permissions, intents, and runtime logs in addition to network traffic data. It also has the following drawbacks: Increase in complexity, large size and dimensionality, potential data imbalance and noise, and less standardized usage than CICAndMal2017 [29].

The Android Ransomware Detection dataset (created by the Canadian Institute of Cybersecurity (CIC) and published on Kaggle in 2023 [17]) is a type of network traffic-based dataset created to identify ransomware activities against Android devices and implemented using machine learning methods. It has 392,035 records (348944 of them were ransomware and 43,091 of them were benign) and 85 features, which are extracted based on the network traffic flows using network monitoring tools (tcp dump, Wireshark, and network gateway monitors). These features are categorized under 13 in Table 1. The data set contains 10 ransomware families, as shown in Table 2, and benign traffic and therefore can be used in binary as well as multi-class classification. Noticeably, the dataset does not contain any missing values, which eases processing and model training. with the following drawbacks: Only network traffic features, limited to 10 ransomware families and are not well generalizable to modern or obfuscated ransomware.

The MH-1M dataset is a large and contemporary Android malware dataset that was released in 2025-2026 to overcome the drawbacks of other datasets, e.g., size, old data, and feature range. It holds about 1.34 million Android applications (APKs) that were gathered during a span of 14 years (2010-2024). The data combines multi-feature representations such as API calls, permissions, intents, and opcodes and rich metadata consisting of file hashes, package names, and Virus Total labeling. Having over 22,000 features and 400+ GB of data, MH-1M is regarded as one of the most comprehensive to conduct research on machine learning, deep learning, and large language model-based malware detection. Its drawbacks include high computational cost, imbalance of classes, fixed preprocessing on its end, complexities of preprocessing, and its potential to contain label noise [30].

Table 1 Categories and data types of features in android ransomware dataset

No	Category	Features Name (Data Types)
1	Flow Identification Features	Flow ID (string), Source IP(string), Source Port(int), Destination IP(string), Destination Port(int), Protocol(int), Timestamp(string).
2	Flow Duration Features	Flow Duration(int).
3	Packet Count Features	Total Fwd Packets(int), Total Backward Packets(int), Subflow Fwd Packets(int), Subflow Bwd Packets(int).
4	Packet Size Features	Total Length of Fwd Packets(int), Total Length of Bwd Packets(int), Fwd Packet Length Max(int),Fwd Packet Length Min(int), Fwd Packet Length Mean(float),Fwd Packet Length Std(float), Bwd Packet Length Max(int),Bwd Packet Length Min(int), Bwd Packet Length Mean(float),Bwd Packet Length Std(float), Min Packet Length(int), Max Packet Length(int), Packet Length Mean(float),Packet Length Std(float), Packet Length Variance(float),Average Packet Size(float).
5	Traffic Rate Features	Flow Bytes/s(float), Flow Packets/s(float), Fwd Packets/s(float), Bwd Packets/s(float), Down/Up Ratio(int).

 ISSN:2302-8149
e-ISSN:2540-9719

6	Inter-Arrival Time (IAT) Features	Flow IAT Mean(float), Flow IAT Std(float), Flow IAT Max(int), Flow IAT Min(int), Fwd IAT Total(int), Fwd IAT Mean(float), Fwd IAT Std(float), Fwd IAT Max(int), Fwd IAT Min(int), Bwd IAT Total(int), Bwd IAT Mean(float), Bwd IAT Std(float), Bwd IAT Max(int), Bwd IAT Min(int).
7	TCP Flag Features	FIN Flag Count(int), SYN Flag Count(int), RST Flag Count(int), PSH Flag Count(int), ACK Flag Count(int), URG Flag Count(int), CWE Flag Count(int), ECE Flag Count(int), Flags(int), Fwd PSH Flags(int), Bwd PSH Flags(int), Fwd URG Flags(int), Bwd URG Flags(int).
8	Header Features	Fwd Header Length(int), Bwd Header Length(int), Fwd Header Length.1(int).
9	Bulk Transfer Features	Fwd Avg Bytes/Bulk(int), Fwd Avg Packets/Bulk(int), Fwd Avg Bulk Rate(int), Bwd Avg Bytes/Bulk(int), Bwd Avg Packets/Bulk(int), Bwd Avg Bulk Rate(int).
10	Window Size Features	Init_Win_bytes_forward(int), Init_Win_bytes_backward(int).
11	Segment Features	act_data_pkt_fwd(int),min_seg_size_forward(int), Avg Fwd Segment Size(float), Avg Bwd Segment Size(float).
12	Flow Activity Features	Active Mean(float),Active Std(float),Active Max(int),Active Min(int), Idle Mean(int),Idle Std(int),Idle Max(int),Idle Min(int).
13	Label Column	Label(string).

Table 2. Types and number of samples of Ransomware dataset

NO.	RANSOMWARE TYPE	NO. OF SAMPLES
1	Svpeng	54161
2	PornDroid	46082
3	Koler	44555
4	RansomBO	39859
5	Charger	39551
6	Simplocker	36340
7	WannaLocker	32701
8	Jisut	25672
9	Lockerpin	25307
10	Pletor	4715

2.4. Feature Selection Methods

In machine learning, feature selection is a crucial preprocessing phase for determining the most informative features and discarding irrelevant ones. It increases the interpretability of the model, decreases its complexity, and prevents overfitting. Feature selection can also improve the machine learning models' overall performance and accuracy [31]. The significance has risen as large data sets and real-time applications have evolved in various areas like healthcare, finance, and bioinformatics [32].

The process of feature selection plays an important role in the ransomware detection system because it can identify malicious activities and differentiate them from benign activities. The selection of relevant features is done manually using feature engineering and expert knowledge [33]. SHAP (Shapley Additive Explanations) is a popular approach to explaining predictions made by machine learning models [34]. Based on game theory, SHAP offers a way to explain model behavior by assigning importance scores to each of the features [35]. SHAP is used in feature selection by using a model to determine the significance of features in predicting outcomes [36].

 ISSN:2302-8149
e-ISSN:2540-9719

2.5. Ransomware Detection for Android

Researchers have suggested a number of methods to detect ransomware targeting the Android operating system, which can be broadly divided into two categories based on data analysis methods: static and dynamic analysis. Static analysis is an examination of the code of the application without running the program and is a standard technique used by antivirus developers [17]. This method allows understanding the structural elements of the application and helps verify compliance with industry standards through the analysis of such aspects as Android permissions, API calls, the Manifest file, and Opcode sequences. Nonetheless, cybercriminals have developed methods like polymorphism, obfuscation, packing, encryption and update attacks to avoid detection systems [37]. The method of the static analysis is comparatively quick and safe, and it can give detailed information regarding malware samples. Nevertheless, it has a number of limitations too [3]. Dynamic analysis, also referred to as behavior-based analysis, involves examining features extracted during the execution of an application. This approach relies on information gathered at runtime—such as system calls, file system behavior, CPU and memory usage, and network activity—within the operating system to detect malicious behavior. Among these, capturing network traffic is considered a critical aspect of effective malware analysis [38]. Both static and dynamic analysis techniques offer distinct advantages and limitations. Static analysis is generally faster and safer, as it does not require code execution; however, its effectiveness can be compromised by obfuscation techniques employed by malicious software to hide harmful behavior. In contrast, dynamic analysis—conducted during program execution—provides greater resilience against evasion tactics such as polymorphism and code obfuscation, as it observes the malware's behavior in real time [17]. Hybrid analysis represents an integrated approach that combines features from both static and dynamic analysis methods. By leveraging the strengths of static analysis—examining the application without execution—and dynamic analysis—monitoring behavior during runtime—this technique enhances the overall effectiveness of malware detection. It aims to improve the accuracy and reliability of threat identification, particularly in detecting sensitive information, thereby offering more robust and comprehensive protection [39].

3. Literature Review

Ransomware detection has become one of the most important concerns in cybersecurity research. Numerous alternative approaches have been proposed over the years beginning with the older signature based approaches to more advanced machine learning and behaviour based detection systems. Various research works resulted in development of Intrusion Detection System (IDS) models to detect ransomware attacks as explained below. In [40], the authors suggested a deep learning-based ransomware detection approach in an Android platform through a Long Short-Term Memory (LSTM) model. The pre-processing phase employed eight feature selection tools using the WEKA tool, with the results being filtered to 19 most informative features. The authors relied on the total CAndMal2017 dataset, which has 1509550 records (1048574 benign and 460976 ransomware cases) that has been divided into training and testing samples (80:20). The evaluation of the model was based on different metrics, and the outcome was that the model had an accuracy of 97.08, a precision of 97, a recall of 97, an F1-score of 97, and an AUC of 0.96. The authors assume that the suggested model can be efficient, scalable, and capable of detecting ransomware in the instances of its implementation at the kernel level of the Android operating system. In [41], the authors presented a novel Android ransomware detection method based on network traffic analysis. To choose relevant traffic features in the CICAndMal2017 dataset, the approach uses the Particle Swarm Optimization (PSO) algorithm. Network traffic is then classified using Decision Tree and Random Forest classifiers based on the filtered features. The suggested method's efficacy was assessed in two different scenarios: detecting ransomware traffic and distinguishing particular ransomware types from benign traffic. According to the results, the Random Forest classifier performs better than the others in detecting ransomware in general with accuracy 81.58%, but the Decision Tree classifier distinguishes certain ransomware kinds with more accuracy (68.94%). Accuracy improvements of 2.26% and 3.7% were observed in the first and second scenarios, respectively. Additionally, the feature selection process significantly reduced the feature set by 56.01% to 91.95%. The optimization process demonstrated rapid convergence, reaching optimal performance within approximately ten iterations. In [42], the authors

 ISSN:2302-8149
e-ISSN:2540-9719

created effective, accurate, and reliable models of binary classification based on ML- and DL-based models. The publicly accessible dataset (named Android Ransomware) that was utilized in training and testing the models consisted of 392,035 records of benign traffic and 10 types of Android ransomware attacks. Two experiments were performed using DT, SVM, KNN, an ensemble model of DT, SVM, and KNN, FNN, and TabNet. Experiment 1 utilized all seventy features. The top 19 features were utilized in Experiment 2. The evaluation of these models was conducted regarding their accuracy, precision, recall, and F1-score. As a result, DT worked more effectively, showing an F1 score of 98.45, an accuracy of 97.24, and a precision of 98.50. The SVM model gave the highest recall of 100%. In [43], the author has suggested an anomaly detection system based on machine learning to detect ransomware in Android network flows. The experiment involved a large labeled dataset of benign traffic and another family of 10 ransomware types referred to as Android ransomware with each sample consisting of 86 network flow features. The model of deep neural network with more than one hidden layers was created and trained using the backpropagation algorithm, where the activation functions used in the hidden layers were ReLU and the activation function used in the output was a sigmoid. A significant amount of preprocessing such as data cleaning, missing values, feature scaling, and categorical encoding was done on the dataset and then divided into three parts, training (70%), validation (15%), and testing (15%). Accuracy, precision, recall and F1-score are some of the standard measures that have been used to evaluate the model. In this case, the results of the experiments were excellent, with the accuracy varying between 99.3% and 99.9, precision between 95.8% and 99.7, recall between 97.3 and 99.9, as well as F1-scores between 96.5 to 99.8 in different classes. Such high values imply that there are low false positive and false negative values. The suggested model demonstrates effectiveness of deep learning methods in mobile network security and proposes the future performance enhancements, including real-time detection and connection with overall cybersecurity structures. In [33], The authors have suggested an Android ransomware detection method applied on the feedforward neural network (Multilayer Perceptron, MLP) using Keras. It is based on three layers of neurons (64, 64, and 2) with ReLU and softmax activation that automatically learn behavioural patterns of ransomware without manually engineered features. To assess it, the authors created a dataset based on two sources, AndroZoo (benign samples) and RansomProber (malicious samples). The dataset also has 54234 APK samples (27117 ransomwares and 27117 benign) with 16 features extracted and, as such, 80 percent training and 20 percent test. The given model performed well, with an accuracy of 98.9, a recall of 1.0, a precision of 0.5, and an F1-score of 0.662. The findings suggest that the model is very efficient in identifying the instances of ransomware, especially with all the positive cases, but with moderate accuracy. On the whole, it can be concluded that the study proves MLP-based deep learning useful in detecting Android ransomware. In [44], the authors suggested an ensemble machine learning system Android ransomware detection based on the use of various classifiers, including Bagging, Random Forest, XGBoost, AdaBoost, Gradient Boost, and CatBoost. The suggested system was based on a huge dataset with 203,556 network traffic records of 10 ransomware families, and non-ransomware traffic, and 85 extracted features, which are the network behavior traits. The model included extensive preprocessing, feature selection with the help of the Random Forest, and extensive cross-validation to improve the efficiency and strength of the model. The experimental findings showed outstanding performance, especially through the Bagging ensemble classifier. The model had an almost perfect classification performance, all accuracy, precision, recall, and F1-score were over 99% in both binary and multi-class classification. In the detailed assessments, some ransomware types were able to reach 100% accuracy, precision, recall, and F1-score, and other types of ransomware were around 99.95%–99.99% which are very low false alarms. In addition to that, comparative analysis revealed that the proposed ensemble approach performed much better than the traditional machine learning models with 100% accuracy in binary classification and about 99.82 in multi-class classification with 99.73 F1-score. These findings draw the attention to the efficiency, scalability, and strength of ensemble learning methods to identify various ransomware families within Android systems. In [45], the authors presented a machine learning-based framework to detect and classify Android ransomware families based on network behavioral analysis rather than static or dynamic methods. The method first derives 84 features of the network traffic, and then best-first search with wrapper evaluation is applied to narrow down to 10 discriminative features and then grouped into four clusters using k-means

 ISSN:2302-8149
e-ISSN:2540-9719

clustering. The framework is tested using a large ransomware network traffic data set of 392,035 samples, which contains 348,944 ransomware samples and 43,091 benign samples, and 10 ransomware network traffic families (e.g., SVpeng, PornDroid, Koler, and Lockerpin). As it has been demonstrated in experiments, Random Forest (RF) is the best-performing classifier in all the experiments. Combining the processes of feature selection and clustering, RF achieves the accuracy of 95.21, 95.27, 95.31, and 95.24 for recall and precision and F1-score, respectively. In addition, the results indicate that feature reduction reduces convergence time by around 78%, whereas clustering results in only small overheads (68) but provides a stronger classification. In general, the framework reveals high effectiveness in properly identifying and separating ransomware families based on the network traffic properties. In [29], the authors suggested a deep learning model to detect Android ransomware using both Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN). The proposed model uses a hybrid feature model, which combines both static features (permissions and intents) and dynamic features (API calls), and allows better representation of application behaviour. The experiments were performed on the CICInvesAndMal2019 (CICAndMal2019) dataset, which includes 426 malware samples and 5065 benign applications with several ransomware families, including WannaLocker, Simplocker, and Lockerpin. The measurement was based on 10-fold cross-validation and common measures such as accuracy, precision, recall, and F1-score. The results indicate a high level of performance of the two models. The CNN model performed the highest at 40 epochs with an accuracy of 97.93, a precision of 98.00, a recall of 99.93, and an F1-score of 98.95. Meanwhile, the LSTM model reached the best performance after 10 epochs, and the accuracy, precision, recall, and F1-score were 97.74, 97.74, 100, and 98.86, respectively. Altogether, CNN marginally beat LSTM in accuracy, whereas the latter exhibited a quicker convergence and recall. These results suggest the usefulness of deep learning and CNN, in particular, in detecting Android ransomware as accurately as possible. In [16], the authors suggested a multi-stage machine learning architecture of Android ransomware detection that emphasizes behavioural profiling and feature reduction as a way of mitigating the issues of high-dimensional feature space and computational complexity. This method derives behavioural characteristics (e.g., systems activity patterns) and uses feature selection methods to eliminate redundancy without compromising detection performance. A number of classifiers are tested, such as Random Forest, Gradient Boosting, Extra Trees, K-Nearest Neighbors, Decision Tree, and CatBoost. The model is tested on a proposed dataset of 3,346 samples (1,673 ransomware and 1,673 benign applications) and the reduced set of features (35 selected features). According to experimental findings, ensemble techniques perform better than alternative classifiers, with Extra Trees having the highest overall performance, best accuracy, best precision, best recall, and best F1-score, with an ROC-AUC of 0.993. The competitive performance of other ensemble models like Random Forest and Gradient Boosting also proves the efficiency of the provided feature reduction strategy to enhance accuracy in detection and lower computational cost. Altogether, the analyzed papers indicate that the Android ransomware detection process is largely developed in terms of both machine learning and deep learning methods, with numerous solutions showing great accuracy and good classification rates in various datasets. There is an evident tendency towards the utilization of more sophisticated models, including LSTM, CNN, and ensemble learning approaches, which frequently get complemented with efficient feature selection and reduction techniques to increase efficiency and scalability. Also, analyzing network traffic and hybrid feature representations has been shown to be a promising solution to enhance detection capability. Nevertheless, even with these developments, there are still issues regarding generalization between datasets, imbalanced data processing, minimizing the number of false positives, and real-time implementation in real-world settings. Consequently, additional studies are necessary to create other better, lightweight, and adaptable models that will be able to support high detection performance and overcome these weaknesses in real-world Android systems. Thus, this paper extends these findings by developing a network traffic-based ransomware detection system that is based on an optimal choice of features, dataset balancing, and effective machine learning models to reach high detection rates and low computational complexity that can be implemented in the real-life Android context.

 ISSN:2302-8149
e-ISSN:2540-9719

4. Proposed Method

To enhance the performance of machine learning algorithms in the existing literature, this paper proposed a lightweight model that has the ability to detect and classify network traffic as ransomware or benign by utilizing traditional and modern machine learning algorithms with a very recent dataset. At the pre-processing stage, the useful features were selected from the dataset, the dataset was balanced by generating new data similar to real data, and the best normalization method was used to enhance the performance of the detection algorithms as much as possible. Data gathering, pre-processing, feature selection, classification, and evaluation were the main steps in the consistent methodology used to perform this research, as illustrated in Figure 1. The five steps can be briefly explained as follows:

A. Data acquisition: Gathering a network traffic feature dataset using a traffic capture tool such as CICFlowMeter [46] or from a website such as Kaggle [47]. The Android Ransomware Detection dataset was used in this paper due to the following characteristics: Large, well-labeled, multi-modal (static + dynamic + network), and up to date.

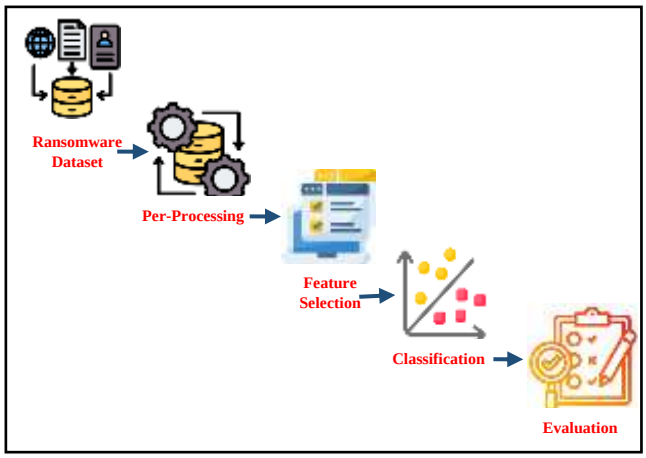


Figure 1 Steps followed to build the proposed model

B. Data Pre-processing: Pre-processing of data is a very important step in the development of effective Android ransomware detection systems. Raw data, be it extracted out of APK files, network traffic, or live analysis, is usually noisy, inconsistent, and high-dimensional and thus cannot be directly applied in a machine learning model. Preprocessing converts this raw data to a clean, structured, and meaningful format that enhances model accuracy, efficiency, and reliability. The first steps to learn about the distribution of the classes and locate skewness are the preprocessing and exploration of the dataset. Android Ransomware.csv dataset is built up of 85 features and 392035 records of network traffic. This is an imbalanced dataset (384944 of them were ransomware and 43,091 of them were benign); therefore, it must contain the means to mitigate the imbalance. Some approaches to reducing imbalance include data-level (e.g., oversampling of minority classes and undersampling of majority classes) and algorithm-level (e.g., ensemble methods and GAN algorithms). GAN is chosen to produce synthetic ransomware and benign samples similar to real samples, which helps to balance the classes and increase the model's generalization. This is particularly important due to the limited diversity of ransomware and benign samples in the dataset. Table 3 shows the architecture and hyperparameters for GAN algorithm. Then, dataset cleaning by removing the missing value in the columns and removing features with low variance values. The features with values that differ slightly or that are the same in each row of the column (i.e., there are zero values in all rows in the column) will not contribute to the informative power of the model and will impose an unwarranted computational cost. Four columns (Flow ID, Source IP, Destination IP, and Timestamp) were removed from the data frame that was entered into the ML algorithms because their values are strings and cannot be used for training the algorithms. Next, the technique of data normalization was applied to scale and modify the data in the range [0 , 1]. A data normalization method is applied to make sure that every feature has an equal role to play in the

 ISSN:2302-8149
e-ISSN:2540-9719

learning process. In this paper, the Z-score normalization (standardization) method and the robust scaler method are used. Finally, label encoding.

Table 3 Summary of the architecture, and training hyperparameters applied to GAN model.

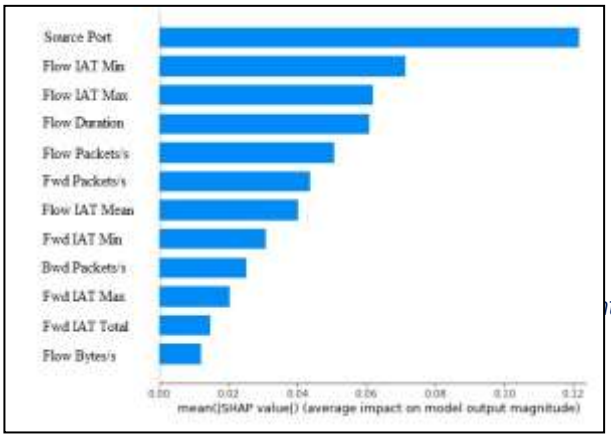
GENERATOR		DISCRIMINATOR	
PARAMETER	**VALUE**	**PARAMETER**	**VALUE**
INPUT NOISE SIZE	100	Input Features	85
HIDDEN LAYERS	2 (128, 256)	Hidden Layers	2 (256, 128)
ACTIVATION FUNCTION	ReLU	Activation Function	LeakyReLU (α = 0.2)
OUTPUT ACTIVATION	Tanh	Output Activation	Sigmoid

C. Feature Selection: Feature selection is one of the most significant processes in the creation of an effective ransomware detection system. Android ransomware datasets have a large number of features, such as permissions, API calls, intents, system calls, network activities, and dynamic behavioral traces. Although these features are useful, not every single one of them brings the same value to the process of recognizing ransomware. The irrelevant or redundant features may make the computation more complex, slow down the training of the model, and even decrease the detection accuracy. The feature selection is an approach that allows the identification of the most relevant and informative features that can be used to differentiate between ransomware and benign programs. The feature selection contributes to the better performance of the models, increases generalization, and reduces the overfitting possibility by lowering the dimensionality of the dataset. It also causes the detection system to be more interpretable and efficient, and this is critical with real-time mobile security applications. This paper utilized a Shapley Additive explanation (SHAP) and Random Forest (RF) to analyze and select the best features from the dataset. Table 4. shows the important feature value (14 features) resulting from applying the Random Forest Classifier on the dataset with the importance value threshold (0.01).

Table 4 Importance features using random forest

NO.	FEATURE NAME	IMPORTANCE VALUE
1	Source Port	0.060162
2	Flow IAT Min	0.045849
3	Flow IAT Max	0.043640
4	Flow Duration	0.042854
5	Flow Packets/s	0.041778
6	Fwd Packets/s	0.041755
7	Flow IAT Mean	0.041292
8	Init_Win_bytes_forward	0.034862
9	Fwd IAT Min	0.029980
10	Bwd Packets/s	0.027769
11	Fwd IAT Max	0.025970
12	Fwd IAT Total	0.025717
13	Fwd IAT Mean	0.025030
14	Flow Bytes/s	0.020994

While Figure 2. shows the important feature value (12 features) resulting from applying the Shapley Additive exPlanation method on the dataset.



 ISSN:2302-8149
e-ISSN:2540-9719

Figure 2 Feature selection using the SHAP method

D. Classification: modern and traditional machine learning algorithms are usually used to perform classification and have been found to be effective, interpretable, and computationally efficient in addressing various cybersecurity tasks. When applied to the Android ransomware detection, these algorithms are trained on labeled data, i.e., the outputs are labeled as either malicious (ransomware) or benign. When trained, the model is able to make predictions about the classification of new, untried applications. In this paper, the dataset was split 80:20 between a training set and a testing set to ensure a fair and unbiased evaluation. The test set was always kept separate and not used in the training of the models, feature selection, or data augmentation. Only the training set was used for steps that could introduce bias, like class balancing and data augmentation. To mitigate class imbalance, class imbalance was addressed using the GAN method and oversampling techniques on exclusively the training data after the train–test split. This way, any synthetic or duplicated samples won't be present in both the training and test sets, so there is no data leakage. The test set did not change and was representative of the real-world data distribution during the evaluation process. Furthermore, feature selection techniques were applied to the training set only - Random Forest importance and SHAP - and this subset of features was used for both training and test sets. This guarantees that the information contained in the test set does not affect the training of the model and the selection of features. This paper focuses on the application of eight modern and traditional classification algorithms (Logistic Regression (LR), Multi-Layer Perceptron (MLP), Decision Tree (DT), K-Nearest Neighbors (KNN), Tabular Networks (TabNet), Support Vector Machine (SVM), Deep Neural Network (DNN), and Convolutional Neural Network (CNN)) to Android ransomware datasets. A baseline deep learning model (DNN) is used to capture the non-linear relationship between features. It is much simpler to be computed than CNN, and comparison is possible between structured learning (DNN) and feature-extracting architectures (CNN). Table 5 shows the architecture and hyperparameters for DNN and CNN algorithms. In addition, it will compare and contrast various models or approaches, thus establishing the most effective and reliable method of ransomware detection. Finally, this step is critical in building the resilient mobile security systems that can detect and overcome the ransomware threats in the real world.

Table 5 is a summary of the architecture, and training hyperparameters applied to CNN and DNN models

CNN		DNN	
PARAMETER	**VALUE**	**PARAMETER**	**VALUE**
INPUT FEATURES	Selected Features	Input Features	Selected Features
CONV LAYERS	2 (32, 64 filters; kernel size = 3)	Hidden Layers	3 (128, 64, 32)
ACTIVATION FUNCTION	ReLU	Activation Function	ReLU
POOLING	MaxPooling (pool size = 2)	Output Activation	Sigmoid
DENSE LAYER	128	Optimizer	Adam
DROPOUT	0.5	Learning Rate	0.001
OUTPUT ACTIVATION	Sigmoid	Batch Size	32
OPTIMIZER	Adam	Epoch	30
LEARNING RATE	0.001	Validation Split	0.2
BATCH SIZE	32	Loss Function	Binary Crossentropy

 ISSN:2302-8149
e-ISSN:2540-9719

EPOCHS	20-35
LOSS FUNCTION	Binary Crossentropy

E. Evaluation: The evaluation is the most important step after the data preprocessing, feature selection, and classification steps when using traditional machine learning algorithms. This measure identifies the performance of the trained model to identify the distinction between ransomware and benign applications and to determine its usability in practice. This is necessary since overall accuracy is not enough to evaluate performance, particularly in Android ransomware datasets, where the problem of class imbalance is prevalent. Thus, to offer a complete analysis of the model performance, several evaluation metrics are employed. *Accuracy* is a metric that determines the percentage of correctly identified cases (ransomware and benign) within the entire dataset. Accuracy, though it gives us a general view of the data, may not be accurate in the imbalanced datasets whereby a model may prefer the majority class. *Precision* measures the number of samples that are considered ransomware that actually are ransomware. A low false positive rate would be reflected by high precision, which would be important to prevent false alarms to users or legitimate applications being blocked. *Recall (Sensitivity)* is the measure of how well the model is able to identify ransomware samples. This measure is essential to cybersecurity since a failure to detect a ransomware example (false negative) may result in severe consequences (loss of information or financial harm). The *F1-score* is the harmonic mean of the precision and the recall, which gives a balanced assessment between the two. It is especially effective when balancing false positives and false negatives is required, in the case of skewed datasets. The *confusion matrix* is a table that displays the number of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). It gives a very easy and precise breakdown of model predictions, and this enables researchers to know the nature of errors being committed. This matrix is essential in the derivation of all other assessment metrics and model weakness diagnosis. The *ROC curve* is a graph-based representation that shows the trade-off between the true positive rate (recall) and the false positive rate. The *Area Under the Curve (AUC)* is used to summarize the ROC curve into one value between 0 and 1. A model whose AUC is nearer to 1 suggests a high degree of discrimination between ransomware and benign classes, whereas when it is near 0.5, the model is performing at random. In this paper, the evaluation step utilizes these metrics in evaluating the effectiveness of the traditional machine learning algorithms in the detection of Android ransomware. Accuracy, precision, recall, F1-score, ROC analysis, AUC, and confusion matrix evaluation help to obtain a full picture of model performance. Additionally, the time for training and testing stages was computed; in all the detection models, the time factor plays a big role in reducing the risks of attack. This multi-metric test is taken to make sure that the proposed detection system is not just accurate but also reliable and robust and able to cope with real-world issues like class imbalance and changing ransomware tactics.

5. Experiments and Results

A series of experiments were implemented during the training and testing of the proposed models under different conditions. These experiments were planned with a number of important criteria, which include the data normalization method used, the dimensionality of the input feature space defined by feature importance scores based on feature selection mechanisms, and algorithm-specific hyperparameters. The scikit-learn tool GridSearchCV is used to automatically search the space of hyperparameters by trying every possible combination in a grid and cross-validating them.

In the first group of experiments, a random Forest (RF) feature selection method was used to generate feature subsets of different sizes, namely 64, 45, 25, and 15 features (including the label). Table 6 describes the results of these experiments using the imbalanced dataset of 348,944 ransomware samples and 43,091 benign network traffic samples. This group used the Standard Scaler normalization method of data preprocessing.

The results of the second group of experiments, where the classification algorithms were applied but the experimental conditions were much different, are presented in Table 7. In the current case, the Shapley Additive Explanations (SHAP) approach was applied to feature selection, which led to the selection of a smaller subset of features, 13 features (12 features and the target label, the reduction rate is about 86%). Moreover, the issue of class imbalance was solved with the help of oversampling

 ISSN:2302-8149
e-ISSN:2540-9719

of the minority class, which provided balanced data with 348,944 samples per class. The Robust Scaler was used to carry out data normalization and is better adapted to outliers.

Although the results of the experiments in Table 6 are quite acceptable, these experiments are limited by a number of factors. The dependence on a lopsided dataset, the application of a conventional normalization method, and the incorporation of a high-dimensional feature space all place significant computational load in terms of processing time, memory usage, and storage demands. These features make these configurations unsuitable to use in resource-constrained environments, including Internet of Things (IoT) devices, where a lightweight and efficient malware detection system is needed.

Table 6 Group 1 experiments results

EXP. NO.	ALGORITHM	PARAMETERS	NO. OF FEATURES	METRICS			TIME IN SECONDS
1	Logistic Regression (LR)	Epoch=1000	64	Accuracy=90 f1-score=89	Recall= 90	Precision=88	24.18
			45	Accuracy=89 f1-score=88	Recall= 89	Precision=87	23.27
			25	Accuracy=89 f1-score=87	Recall= 89	Precision=87	9.16
			15	Accuracy=88 f1-score=86	Recall= 88	Precision=85	3.52
2	Multi-Layer Perceptron (MLP)	No. of hidden layer =100 Epoch=300 Activation function=Rule Solver-Adam	64	Accuracy=89 f1-score=89	Recall= 89	Precision=90	127.67
			45	Accuracy=89 f1-score=88	Recall= 89	Precision=79	115.2
			25	Accuracy=89 f1-score=87	Recall= 89	Precision=85	110.64
			15	Accuracy=89 f1-score=86	Recall= 89	Precision=88	96.67
3	Decision Tree (DT)	Criterion=gini Max_depth=5	64	Accuracy=91 f1-score=89	Recall= 91	Precision=91	7.51
			45	Accuracy=91 f1-score=89	Recall= 91	Precision=91	6.70
			25	Accuracy=91 f1-score=89	Recall= 91	Precision=91	5.35
			15	Accuracy=91 f1-score=89	Recall= 91	Precision=90	3.73
4	KNN	n_neighbors=5	64	Accuracy=93 f1-score=93	Recall= 93	Precision=92.7	120
			45	Accuracy=92.8 f1-score=93	Recall= 93	Precision=92	103
			25	Accuracy=92.9 f1-score=92	Recall= 92	Precision=92	82.54
			15	Accuracy=92.1 f1-score=92	Recall= 92	Precision=92	21.98
5	TabNet	max_epochs=10 patience=10 batch_size=256 num_workers=0	64	Accuracy=91 f1-score=89	Recall= 91	Precision=90	843
			45	Accuracy=90 f1-score=88	Recall= 91	Precision=90	725
			25	Accuracy=90.5 f1-score=89	Recall= 91	Precision=89	598
			15	Accuracy=90 f1-score=89	Recall= 91	Precision=89	557
6	Support Vector Machin (SVM)	Kernel = rbf C=1.0 Gamma= scale	64	Accuracy=90.3. f1-score=90	Recall= 91	Precision=90.6	5733
			45	Accuracy=90 f1-score=88	Recall= 91	Precision=90	5371
			25	Accuracy=90 f1-score=88	Recall= 91	Precision=90	4335
			15	Accuracy=89.8 f1-score=86	Recall= 90	Precision=89	4850
7	Deep Neural Network (DNN)	epochs=30 batch_size=32 validation_split=0.2 activation='sigmoid'	64	Accuracy=92 f1-score=92	Recall= 92	Precision=92	358
			45	Accuracy=91 f1-score=91	Recall= 92	Precision=91	343
			25	Accuracy=91.7 f1-score=91	Recall= 92	Precision=91	343
			15	Accuracy=91 f1-score=90	Recall= 91	Precision=90	338
8	Convolutional Neural Network (CNN)	Optimizer: Adam Learning Rate: 0.001 Batch Size: 32 Epochs: 20–30 Loss Function: Binary Crossentropy	64	Accuracy=94 f1-score=94	Recall= 93	Precision=94.8	365
			45	Accuracy=95.2 f1-score=94	Recall= 93	Precision=94.7	329
			25	Accuracy=94 f1-score=94	Recall= 93	Precision=94.6	296
			15	Accuracy=95.6 f1-score=94	Recall= 93	Precision=94.6	276

 ISSN:2302-8149
e-ISSN:2540-9719

Conversely, Table 7 shows that the experimental results are significantly better in terms of both detection performance and computational efficiency. Such improvements can be explained by several factors, such as the dimensionality of features, which in turn reduces the processing time, directly leading to quicker detection and thus less possibility of data compromise. Also, the implementation of the Robust Scaler enhances data representation much better by reducing the effect of outliers, which results in more reliable input to the classification algorithms.

It is worth noting that the Support Vector Machine (SVM) classifier shows excellent results on several evaluation metrics in Table 7. A high accuracy also means that it has a strong classification capacity, whereas the high precision means that it is good at detecting real positive samples with a low number of false positives. The recall measure also illustrates how the model is capable of identifying real ransomware traffic, such as minor or less recognizable patterns. Lastly, the F1-score reflects the overall stability and performance of the model in detecting ransomware against other data types, which is proven by the balancing relationship between precision and recall.

Table 7 Group 2 experiments result

EXP. NO.	ALGORITHM	PARAMETERS	NO. OF FEATURES	METRICS			TIME IN SECONDS
1	Logistic Regression (LR)	Epoch=1000	12	Accuracy=93 f1-score=91.21	Recall= 91.43	Precision=91	2.12
2	Multi-Layer Perceptron (MLP)	No. of hidden layer =100 Epoch=300 Activation function=Rule Solver-Adam	12	Accuracy=93.41 f1-score=91.5	Recall= 92	Precision=91.01	65.32
3	Decision Tree (DT)	Criterion=gini Max_depth=5	12	Accuracy=96.09 f1-score=96.61	Recall=97.16	Precision=96.03	2.33
4	KNN	n_neighbors=5	12	Accuracy=97.51 f1-score=97.2	Recall= 97.32	Precision=97.1	16.33
5	TabNet	max_epochs=10 patience=10 batch_size=256 num_workers=0	12	Accuracy=94.02 f1-score=95.42	Recall=95.89	Precision=94.96	213
6	Support Vector Machin (SVM)	Kernel = rbf C=1.0 Gamma= scale	12	Accuracy=98.40 f1-score=96.88	Recall=97.34	Precision=96.44	113
7	Deep Neural Network (DNN)	epochs=30 batch_size=32 validation_split=0.2 activation='sigmoid'	12	Accuracy=97.91 f1-score=96.56	Recall=96.45	Precision=96.68	209
8	Convolutional Neural Network (CNN)	Optimizer: Adam Learning Rate: 0.001 Batch Size: 32 Epochs: 25 Dropout: 0.5 Loss: Binary Crossentropy	12	Accuracy=97.02 f1-score=97.1	Recall=96.08	Precision=97.05	283

Moreover, as a way to increase the performance of the proposed models, Generative Adversarial Networks (GANs) have risen as a promising technique of data augmentation. GANs are composed of two neural networks, which are a generator and a discriminator, and are trained competitively in such a way that the generated synthetics are similar to real samples. GANs can be used to create realistic ransomware and benign samples, which can be used to balance the dataset and thus enhance the robustness and generalization capabilities of classification models. Therefore, the data augmentation by GAN is combined with conventional classification methods. The proposed framework will initially use GANs to come up with synthetic samples to balance the dataset. The augmented dataset is, furthermore, utilized to train the above classifiers. The integration will be used to improve the accuracy of detection, decrease false negatives, and overall model performance in the detection of Android ransomware. Table 8 shows the results before and after applying the GAN algorithm.

Table 8 The performance before and after GAN augmentation

CLASSIFIER	ACCURACY (BEFORE)	ACCURACY (AFTER)	RECALL	PRECISION	F1-SCORE
LR	93	95.6	94.8	93.9	94.3
MLP	93.41	97.2	96.8	96.3	96.5
DT	96.04	94.1	93.2	92.5	92.8
KNN	97.51	95.0	94.18	94.6	94.3
TABNET	94.02	97.5	97.0	96.5	96.7

 ISSN:2302-8149
e-ISSN:2540-9719

SVM	98.40	96.7	96.1	95.4	95.7
DNN	97.91	98.1	97.6	97.2	97.4
CNN	97.02	99.5	99.8	99.6	99.6

The confusion matrix provides a breakdown of the performance of the classifier model at a detailed level by comparing the actual labels with the predicted labels. It is specifically handy in determining the model effectiveness in distinguishing between benign samples and ransomware samples (i.e., the number of the test samples is 78407, (20% of the dataset)). The confusion matrices of the Convolutional Neural Network (CNN) and GAN applied with CNN are as shown in Table 9 and Table 10, respectively.

Table 9 Confusion matrix for CNN only

ACTUAL \ PREDICTED	POSITIVE	NEGATIVE
POSITIVE	65829	1601
NEGATIVE	735	10242

Table 10 Confusion matrix for (GAN + CNN)

ACTUAL \ PREDICTED	POSITIVE	NEGATIVE
POSITIVE	67296	134
NEGATIVE	241	10736

Once using GAN-based data augmentation, the CNN performance is greatly improved: GAN + CNN model obtained the best accuracy (99.5%) along with a good value of precision, recall, and F1 score. Also, the false negatives decreased to 134, which is a considerable advancement in detecting ransomware. The false positives were also reduced (735 to 241), i.e., the number of benign samples that are labelled as malicious is lower.

The impact of GAN augmentation was not consistent when applied to all models. The CNN and DNN improved their performance, whereas the traditional machine learning algorithms like Decision Tree (DT), K-Nearest Neighbours (KNN), and Support Vector Machine (SVM) suffered from a small performance drop after augmentation. A potential explanation is that, in general, deep learning models can learn more complex feature distributions and can better take advantage of the added variability of synthetic samples. However, noise or distributional shifts in the data created by a GAN may affect the classification model compared to traditional classifiers. For instance, KNN is heavily dependent on the similarity based on distance, and synthetic samples can also change the neighbourhood structures, making classification less reliable. Similarly, the performance of decision tree (DT) models can degrade if the added patterns are noisy, leading to overfitting, and the performance of support vector machine (SVM) models can suffer if the boundary between classes is non-linearly altered in the noise-augmented data and is not optimally separable with the selected kernel configuration.

In conclusion, CNN will greatly benefit through the use of GAN-based data balancing since deep learning models will need enough data to be generalized. The reduction in false negatives is the most important improvement, enhancing system security. The confusion matrix confirms that GAN integration leads to a more robust and reliable detection system.

For more investigation, the CNN model achieves an AUC score (≈0.962) and become (≈0.989) after GAN augmentation, indicating excellent discriminative ability. The trade-off between the false positive rate and the true positive rate of a classifier is presented on the ROC curves. The CNN model demonstrates that ROC performance improves significantly with GAN augmentation, as in the Figure 3.

The experimental findings prove that the combination of GAN and classification models is an effective way to solve the issue of class imbalance. Key findings include:

- Recall and F1-score were significantly improved, particularly in the case of ransomware detection.
- False negatives will be reduced, which improves the security of the system.
- Deep learning models (CNN, DNN) are more effective than traditional machine learning models.
- GAN generated samples enhance model generalization and robustness.

 ISSN:2302-8149
e-ISSN:2540-9719

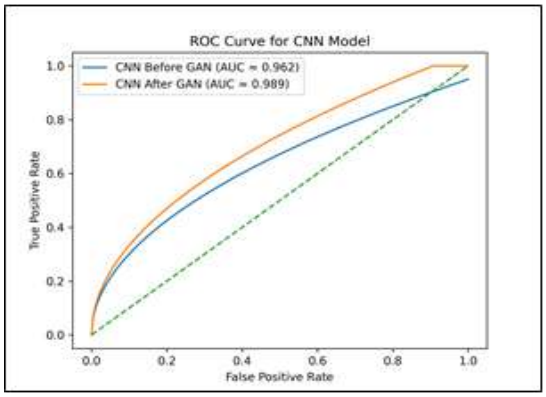


Figure 3 ROC curve for CNN model before and after GAN augmentation

5.1. Comparison to the Existing Methods

To confirm the effectiveness of the proposed GAN-based data augmentation framework, a comparison is made to the current state-of-the-art Android ransomware detection methodologies. Table 11 outlines the main differences in the methodology, working with data, and performance.

Table 11 Compression the best model against existing ransomware detection methods in the literatures review

NO.	RESEARCHERS REF. / YEAR	METHODS	BEST ALGORITHM	DATASET	NO. OF FEATURES	ACCURACY	RECALL	PRECISION	F1-SCORE	AUC
1	[42] / 2023	DT, SVM, KNN, FNN, TabNet	DT Binary classifier	Android Ransomware Imbalanced 392035 samples 85 features	19	97.24	98.40	98.56	98.45	*
2	[43] / 2024	Machine Learning and deep learning	* Multi- classifier	Android Ransomware Imbalanced 392035 samples 85 features	all	99.3	97.3	95.8	96.5	*
3	[44] / 2025	Bagging, RF, XGBoost, AdaBoost, Gradient Boost, and CatBoost	Bagging	Android Ransomware Imbalanced 392035 samples 85 features	all	99.82	*	*	99.73	*
4	[45] / 2025	DT, RF, KNN, SVM, and bagging	RF	Android Ransomware Imbalanced 392035 samples 85 features	10	95.21	95.27	95.31	95.24	*
5	Proposed Model	LR, MLP, DT, KNN, TabNet, SVM, DNN, CNN	SVM	Android_Ransomeware Balanced dataset 392035 instances 12 features	12	98.40	97.34	96.44	96.88	97.43
6	Proposed Model	LR, MLP, DT, KNN, TabNet, SVM, DNN, CNN	GAN+CNN	Android_Ransomeware Balanced dataset 392035 instances 12 features	12	99.5	99.8	99.6	99.6	98.9

6. Conclusion

This paper introduced a lightweight and efficient solution to identify Android ransomware based on network traffic analysis. The approach proposed involved data preprocessing, feature selection with Random Forest and SHAP, data augmentation with the GAN, and testing the performance of multiple classification models. The results showed that GAN + CNN model had the highest overall performance with accuracy, recall, precision, F1 score, and AUC of 99.5%, 99.8%, 99.6%, 99.6%, and 98.9%, respectively. The result shows that feature selection with the augmentation from a GAN could boost the ransomware detection rate and decrease the number of features needed for ransomware classification. The presented framework achieved high accuracy but was tested with a single dataset, which reduces the generalizability of the proposed framework to other ransomware families and real-world scenarios. It was only tested in a lab environment and not in real Android devices or real network traffic, so systemic characteristics, including latency, memory use, and energy consumption, are not confirmed. These are some limitations that future work should include, such as multiple external datasets, real-time deployment scenarios, and cross-validation techniques. Testing

 ISSN:2302-8149
e-ISSN:2540-9719

on various Android environments and operational networks will enhance robustness and scalability assessment as well. Lastly, high-order regularization and robustness techniques should be employed to prevent possible overfitting due to amplified training data.

Acknowledgement

The author would like to thank the university of Mosul for providing the environment to achieve this research.

References

[1] Ribeiro J, Saghezchi FB, Mantas G, Rodriguez J, Abd-Alhameed RA. *Hidroid: Prototyping a Behavioral Host-based Intrusion Detection and Prevention System for Android*. IEEE Access. 2020 Jan 27;8:23154-68. [https://doi.org/10.1109/ACCESS.2020.2969626].

[2] Agrawal P, Trivedi B. *A Survey on Android Malware and Their Detection Techniques*. In2019 IEEE International conference on electrical, computer and communication technologies (ICECCT) 2019 Feb 20 (pp. 1-6). IEEE. [https://doi.org/10.1109/ICECCT.2019.8868951].

[3] Albedwawi S. *ML-based Ransomware Detection for Android Os (Doctoral dissertation,* Khalifa University of Science). 2023. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://khazna.ku.ac.ae/ws/portalfiles/portal/19098217/file.

[4] Ibrahim IM, Sallow AB. *Guarding Android: A Comprehensive Review of Intrusion Detection Techniques for Smartphones*. Science Journal of University of Zakho. 2023 Oct 15;11(4):469-80. [https://doi.org/10.25271/sjuoz.2023.11.4.1161].

[5] Reshmi TR. *Information Security Breaches Due to Ransomware Attacks-a Systematic Literature Review*. International Journal of Information Management Data Insights. 2021 Nov 1;1(2):100013. [https://doi.org/10.1016/j.jjimei.2021.100013].

[6] Wiles A, Colombo F, Mascorro R. *Ransomware Detection using Network Traffic Analysis and Generative Adversarial Networks*. Authorea. September 17, 2024. [https://doi.org/10.22541/au.172659907.77469627/v1].

[7] Aslam N, Steltzer H. *Cybersecurity and Network Security: Strengthening Defenses Against Emerging Threats*. February 2025. [https://doi.org/10.13140/RG.2.2.16350.96321].

[8] Alqahtani H, Sarker IH, Kalim A, Minhaz Hossain SM, Ikhlaq S, Hossain S. *Cyber Intrusion Detection using Machine Learning Classification Techniques*. In International conference on computing science, communication and security 2020 Mar 26 (pp. 121-131). Singapore: Springer Singapore. [https://doi.org/10.1007/978-981-15-6648-6_10].

[9] Elkhadir Z, Chougdali K, Benattou M. *Intrusion Detection System using PCA and Kernel PCA Methods*. InProceedings of the Mediterranean Conference on Information & Communication Technologies 2015: MedCT 2015 Volume 2 2016 Apr 16 (pp. 489-497). Cham: Springer International Publishing. [https://doi.org/10.1007/978-3-319-30298-0_50].

[10] Khraisat A, Gondal I, Vamplew P, Kamruzzaman J. *Survey of Intrusion Detection Systems: Techniques, Datasets and Challenges*. Cybersecurity. 2019 Dec;2(1):1-22. [https://doi.org/10.1186/s42400-019-0038-7].

[11] Arslan RS. *JDroid: Android Malware Detection using Hybrid Opcode Feature Vector*. PeerJ Computer Science. 2025 Jul 25;11:e3051. [https://doi.org/10.7717/peerj-cs.3051].

[12] Zhou H, Yang X, Pan H, Guo W. *An Android Malware Detection Approach based on SIMGRU*. IEEE Access. 2020 Jul 29;8:148404-10. [https://doi.org/10.1109/ACCESS.2020.3007571].

[13] Liu K, Xu S, Xu G, Zhang M, Sun D, Liu H. :*A Review of Android Malware Detection Approaches based on Machine Learning*. IEEE access. 2020 Jul 1;8:124579-607. [https://doi.org/10.1109/ACCESS.2020.3006143].

[14] Chio C, Freeman D. *Machine Learning and Security: Protecting Systems with Data and Algorithms*. 1st ed." O'Reilly Media, Inc."; 2018 Jan 26. https://www.amazon.com/Machine-Learning-Security-Protecting Algorithms/dp/1491979909.

[15] TeckPath Team Lead. *Cyber Security Services in Washington, DC.*, services report. December 20, 2021,: The 2021 Cyber Security Statistics, Data, & Trends. Available: https://purplesec.us/cyber-security-trends-2021/, 2021.

 ISSN:2302-8149
e-ISSN:2540-9719

[16] Sibtain M, Hussain M, Riaz Q, Qadir S, Riaz N, Jung KH. : *Lightweight and Robust Android Ransomware Detection using Behavioral Analysis and Feature Reduction*. Computers, Materials & Continua. 2025 Sep 1;84(3).[https://doi.org/10.32604/cmc.2025.066198].

[17] Ahmed OS, Ibrahim Al-Dabbagh OA. *Ransomware Detection System based on Machine Learning*. Journal of Education & Science. 2021 Nov 1;30(5). [https://doi.org/10.33899/edusj.2021.130760.1173].

[18] Al-doorı SK, Taspınar YS, Koklu M. *Distracted Driving Detection with Machine Learning Methods by CNN based Feature Extraction*. International Journal of Applied Mathematics Electronics and Computers. 2021 Dec 31;9(4):116-21. [https://doi.org/10.18100/ijamec.1035749].

[19] Park HA. *An Introduction to Logistic Regression: from Basic Concepts to Interpretation with Particular Attention to Nursing Domain*. Journal of Korean academy of nursing. 2013 Apr 1;43(2):154-64. [https://doi.org/10.4040/jkan.2013.43.2.154].

[20] Kishore B, Yasar A, Taspinar YS, Kursun R, Cinar I, Shankar VG, et al. *Computer-Aided Multiclass Classification of Corn from Corn Images Integrating Deep Feature Extraction. Computational Intelligence and Neuroscience*. 2022;2022(1):2062944. [https://doi.org/10.1155/2022/2062944].

[21] AlShammari AF. *Implementation of Classification using K-Nearest Neighbors (KNN) in Python.* International Journal of Computer Applications. 2024 ; 975:8887. [https://doi.org/10.5120/ijca2024923894].

[22] Montesinos López OA, Montesinos López A, Crossa J. :*Fundamentals of Artificial Neural Networks and Deep Learning*. InMultivariate statistical machine learning methods for genomic prediction 2022 Jan 14 (pp. 379-425). Cham: Springer International Publishing. [https://doi.org/10.1007/978-3-030-89010-0_10].

[23] Zidan RA, Karraz G. *Towards an Efficient Internet of Things Intrusion Detection by using Support Vector Machine*. Baghdad Science Journal. 2025;22(5):1714-24. [https://doi.org/10.21123/bsj.2024.11067].

[24] McDonnell K, Murphy F, Sheehan B, Masello L, Castignani G. *Deep Learning in Insurance: Accuracy and Model Interpretability using TabNet*. Expert Systems with Applications. 2023 May 1;217:119543. [https://doi.org/ 10.1016/j.eswa.2023.119543].

[25] Kiranyaz S, Avci O, Abdeljaber O, Ince T, Gabbouj M, Inman DJ. *1D Convolutional Neural Networks and Applications: A Survey*. Mechanical systems and signal processing. 2021 Apr 1;151:107398. [https://doi.org/10.48550/arXiv.1905.03554].

[26] Chakraborty, T., Reddy KS, U., Naik, S. M., Panja, M., & Manvitha, B. (2024). *Ten Years of Generative Adversarial Nets (GANs): A Survey of the State-of-the-Art.* Machine Learning: Science and Technology, 5(1), 011001.

[27] Das Y., Laxmi L., Kumar N, Bardhan K. *Ransomware Detection using Artificial Intelligence*. International Journal of Advanced Technology in Engineering and Science, Vol. 11, Nikhil Kumar 2023 May 05 (pp. 1-10). https://www.ijates.com/ADMIN/admin/postimages/images/fullpdf/1683986012_933.pdf.

[28] Alazab M, Khurma RA, Camacho D, Martín A. : *Enhanced Android Ransomware Detection Through Hybrid Simultaneous Swarm-based Optimization*. Cognitive Computation. 2024 Sep;16(5):2154-68. [https://doi.org/10.1007/s12559-024-10301-4].

[29] Abd Rais, N. S., Foozy, C. F. M., & Maslan, A. (2025). *Android Ransomware Detection by Deep Learning*. Journal of Soft Computing and Data Mining, 6(1), 378-393.

[30] Braganca, H., Kreutz, D., Rocha, V., & Assolin, J. (2025). *MH-1M: A 1.34 Million-Sample Comprehensive Multi-Feature Android Malware Dataset for Machine Learning, Deep Learning, Large Language Models, and Threat Intelligence Research*. arXiv preprint arXiv:2511.00342.

[31] Guyon I, Elisseeff A. *An Introduction to Variable and Feature Selection*. Journal of Machine Learning Research. 2003;3(Mar):1157-82. [https://doi.org/10.1162/153244303322753616].

[32] Kraev E, Koseoglu B, Traverso L, Topiwalla M. *Shap-Select: Lightweight Feature Selection using SHAP Values and Regression. arXiv prepr*. arXiv:2410.06815. 2024 Oct 9. [https://doi.org/10.48550/arXiv.2410.06815].

[33] Farhan RI. *An Approach to Android Ransomware Detection using Deep Learning*. Wasit Journal for Pure Sciences. 2024 Mar 30;3(1):90-4. [https://doi.org/10.31185/wjps.325].
[34] Lundberg S, Lee SI. *A Unified Approach to Interpreting Model Predictions*. Advances in Neural Information Processing Systems. 2017;30. [https://doi.org/10.1145/3744333.3747810].
[35] Shapley LS. *A Value for n-Person Games*. 1953 (pp. 307-317). [https://doi.org/10.1515/9781400881970-018].
[36] Wang H, Liang Q, Hancock JT, Khoshgoftaar TM. *Feature Selection Strategies: A Comparative Analysis of SHAP-Value and Importance-based Methods*. Journal of Big Data. 2024 Mar 26;11(1):44. [https://doi.org/10.1186/s40537-024-00905-w].
[37] Saxe J, Sanders H. *Malware Data Science: Attack Detection and Attribution*. No Starch Press; 2018 Sep 25. https://www.amazon.com/Malware-Data-Science-Detection-Attribution/dp/1593278594.
[38] Sihwail R, Omar K, Ariffin KZ. *A Survey on Malware Analysis Techniques: Static, dynamic, Hybrid and Memory Analysis*. Int. J. Adv. SCI. Eng. Inf. Technol. 2018 Sep 30;8(4-2):1662-71. [https://doi.org/10.18517/ijaseit.8.4-2.6827].
[39] Sharma S, Kumar R, Rama Krishna C. *A Survey on Analysis and Detection of Android Ransomware*. Concurrency and Computation: Practice and Experience. 2021 Aug 25;33(16):e6272. [https://doi.org/10.1002/cpe.6272].
[40] Bibi I, Akhunzada A, Malik J, Ahmed G, Raza M. *An Effective Android Ransomware Detection Through Multi-Factor Feature Filtration and Recurrent Neural Network*. In2019 UK/China Emerging Technologies (UCET) 2019 Aug 21 (pp. 1-4). IEEE. [https://doi.org/10.1109/UCET.2019.8881884].
[41] Hossain, M. S., Hasan, N., Samad, M. A., Shakhawat, H. M., Karmoker, J., Ahmed, F., ... & Choi, K. (2022). *Android Ransomware Detection from Traffic Analysis using Metaheuristic Feature Selection*. IEEE Access, 10, 128754-128763.[https://doi.org/10.1109/ACCESS.2022.3227579].
[42] Albin Ahmed, A., Shaahid, A., Alnasser, F., Alfaddagh, S., Binagag, S., & Alqahtani, D. (2023). *Android Ransomware Detection using Supervised Machine Learning Techniques based on Traffic Analysis*. Sensors, 24(1), 189. [https://doi.org/10.3390/s24010189].
[43] Zawaideh, F.H.S. (2024). *Machine Learning-based Anomaly Detection in Android Network Flows for Ransomware Identification*. Global Journal of Information Technology: Emerging Technologies 14(1), 1-13. [https://doi.org/10.18844/gjit.v14i1.9363].
[44] Hossain, M. A., Hasan, T., Ahmed, F., Cheragee, S. H., Kanchan, M. H., & Haque, M. A. (2025). *Towards Superior Android Ransomware Detection: An Ensemble Machine Learning Perspective*. Cyber Security and Applications, 3, 100076. [https://doi.org/10.1016/j.csa.2024.100076].
[45] Singh, M. M., Selvaraj, K., & Wei, Z. (2025). *Enhanced Detection of Android Ransomware Families using Machine Learning and Network Traffic Analysis*. Bulletin of Electrical Engineering and Informatics, 14(4), 2987-2996.[https://doi.org/10.11591/eei.v14i4.9485].
[46] *The CICFlowMeter Packet Capture Tool*. Available: https://www.unb.ca/cic/research/applications.html#CICFlowMeter.
[47] Android Ransmware Dataset form Kaggle Access by https://doi.org/10.34740/kaggle/dsv/4987535," 2023.