ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20220907113227
Original scientific paper

Deep Learning based Malware Detection for Android Systems: A Comparative Analysis

Esra Calik BAYAZIT*, Ozgur Koray SAHINGOZ, Buket DOGAN

Abstract: Nowadays, cyber attackers focus on Android, which is the most popular open-source operating system, as main target by applying some malicious software (malware) to access users' private information, control the device, or harm end-users. To detect Android malware, security experts have offered some learning-based models. In this study, we developed an Android malware detection system that uses different machine\deep learning models by performing both dynamic analyses, in which suspected malware is executed in a safe environment for observing its behaviours, and static analysis, which examines a malware file without any execution on the Android device. The benefits and weaknesses of these models and analyses are described in detail in this comparative study, and directions for future studies are drawn. Experimental results showed that the proposed models gave better results than those in the literature, with 0.988 accuracy for LSTM on static analysis and 0.953 accuracy for CNN-LSTM on dynamic analysis.

Keywords: android; deep learning; malware detection systems; malware analysis

1 INTRODUCTION

Technological developments have shifted from traditional computers to mobile devices, with a 10-times increase in the worldwide market share of mobile devices [1]. With these rapid developments in mobile communication technologies and devices, the Android operating system has become the most preferred operating system for mobile devices due to its open-source structure, accessibility, and scalability advantages [2].

Accordingly, as the number of applications offered by Android systems increases, the number of malwares with more complex code structures also increases day by day, and it becomes difficult to detect. Because of this, these devices need to be checked for and analysed for malware with the help of some software.

One of the main reasons for the increase in Android malware threats is that it is an operating system that can be integrated into all mobile devices, regardless of model. In addition, Android systems are popular because they are an open-source system supported by Google. As mentioned in the "Zimperium Global Mobile Threat Report 2022", the attackers target Android systems. The new Android malware threats increase graph for the 2021 year is shown in Fig. 1 [3].

The applications on Android devices are formed as Application Packages (APK), which contain multiple files and some metadata about the application, such as package name, permissions, static properties, etc. The permissions in this file are an important security mechanism for accessing sensitive resources owned by applications, and they play an active role in limiting these accesses. On the other hand, the application's network activity, file changes, and system calls are dynamic properties. These APKs are the weakest parts of the Android system; therefore, the security mechanism should focus on them.

These APK files are used in third-party app stores that offer apps for Android-based devices as well as official app stores like Google Play. It uses app bundles to build and deliver APKs that are optimized for each device's configuration [6], so they can be easily accessible and downloaded to mobile devices. Additionally, these files can also be loaded into the system manually without third party systems.

These files, which contain particularly malicious code, are stored in the normal application, taking control of a vulnerable system, causing it to perform poorly and intentionally alter its intended function, making malware detection more difficult [5]. Therefore, existing types of malwares, such as Adware, Scareware, Trojans, Ransomware, and Backdoors, easily evade traditional malware methods due to advanced hiding techniques.

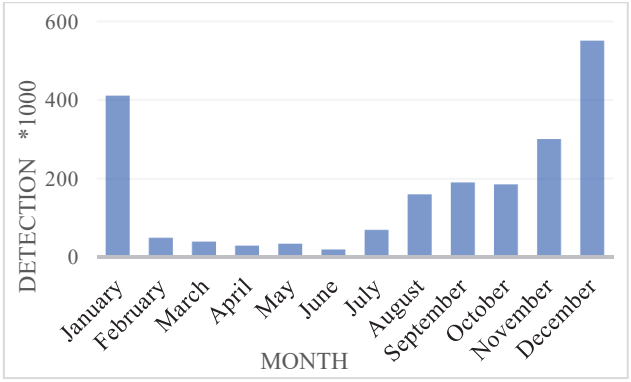

Figure 1 New Android malware threat growth over the 2021

For this reason, Android malware detection studies have been carried out for learning and classification purposes to protect Android devices from malware variants [6-9]. Generally, a malware detection system consists of some detection steps by applying static, dynamic or hybrid analysis technique. The first of these is the signature-based static analysis [6-8, 10] in which the analysis is made before the application is executed in the device. On the other hand, the second one is the dynamic analysis technique [4, 6-9, 11] in which the behaviour is monitored in an isolated environment after the application is run, and the last one is the hybrid analysis technique [4, 7-9] which is using both these analysis techniques. Static analysis studies use permissions, intents, and API calls as attributes for malware detection, while dynamic analysis studies adopt various classification approaches using network flow properties and system calls. Among these techniques, the static analysis technique is preferred more in the literature, especially in terms of cost [8-10]. This is because collecting dynamic attribute data is costly and processing steps are long and difficult. Attackers avoid static, dynamic

and hybrid analysis detection methods by using methods such as encryption, code hiding, packaging [7]. This situation can be overcome by using artificial intelligence methods that provide flexibility and learning capabilities for the software. The rapid growth of Android malware apps and technologies to evade detection systems is rendering traditional defences ineffective. Deep learning takes place in almost every field with its strong feature abstraction ability and has become a prominent research area in recent years. The limited capabilities of machine learning are limiting emerging malware detection systems. First, the amount of data increasing day by day requires the most functional processing and use of system features. Because of this, it is important to show and explain how well static or dynamic analysis techniques work in the malware analysis phase.

In this study, a comparative analysis of dynamic analysis techniques using deep learning in malware detection on Android systems is presented on CIC-AndMal2017 [11] and the second part, static analysis techniques on CIC-InvesAndMal2019 [13] dataset. Thus, by presenting the advantages and disadvantages of both techniques, it is aimed at researchers and practitioners to see the big picture. In the article, the following contributions are suggested:

- Performing static and dynamic analysis of Android
- malware.
- Presenting comparative table of related studies.
- Presenting a comparative analysis of different learning algorithms.
- Using deep learning approaches, present performance results by making a comparison classification of static and dynamic analysis.

The rest of the paper is organized as follows. Section 2 briefly covers a summary of the related works on Android malware detection and identification. Section 3 covers the background of analysis techniques and traditional machine learning and deep learning methods. Section 4 presents our proposed approach for the detection of Android malware and a description of the used data set. In Section 5, data set pre-processing studies and comparative experimental results are explained. Section 6 is concluded with directions for future work.

2 RELATED WORKS

Fast, effective, continuous, efficient, and reliable detection of malicious software in applications on Android devices is an important issue in both the academic, commercial, and industrial worlds. Android operating system is the most preferred operating system in the mobile device industry with a market share of 72% [2]. Therefore, especially malware attacks, are made on these devices to capture the highest number of victims.

Lashkari et al. [11] instead of all the shortcomings and limitations of emulators, developed a new dataset called CICAndMal2017, which includes dynamic features using real smartphones. On the publicly available CICAndMal2017 dataset they created, it showed an average of 85% precision and 88% recall for three classifiers: Random Forest (RF), K-Nearest Neighbour (KNN), and Decision Tree (DT).

The authors of study [12] employed the deep learning-based LSTM algorithm to detect malware on Android. Eight distinct approaches to attribute selection were used to choose features (information gain attribute, gainratio attribute, cvattribute, symmetrical unset attribute, chisquare, onerattribute, relief attribute, and significance attribute). By comparing the outcomes of all feature filtering procedures, the 19 important features were chosen by a simple majority voting process. The CICAndMal2017 data collection was used to detect ransomware in the study. The feature filtration experiment was carried out on WEKA on a total of 40000 samples, with 20000 benign samples and 20000 ransomware-signed samples. According to the findings, the study's accuracy rate was 97%.

In [13], the authors examined features using two-layer Android malware analysis applications on the CICInvesAndMal2019 dataset, which includes permissions and purposes as static features and API calls as dynamic features. The analysis results indicated that the first layer achieved 95% accuracy in static-based malware binary classification, and the second layer achieved 83.3% accuracy in dynamic-based malware category classification.

In the study [14], the detection of the seedling software in the CICAndMal2017 dataset was examined as an experimental study. It has been reported that the random forest classification method achieves the highest success rate among other traditional machine learning algorithms, with an 82.80% success rate in ransomware detection.

Hr et al. [15] presented a study in which they detected malware on Android systems using the static analysis technique. In the study, the dataset created from the applications obtained from the Google Play Store and Virus Share was used. It was stated that a 94.64% accuracy rate was achieved by using DBN as the learning model.

DeepDroid [16] is a framework that consists of three parts. These stages are as follows: data gathering, feature selection, and machine learning. The study analysed 120000 Android applications that make use of API calls and permissions. The study analysed 100000 APK files downloaded from the Google Play Store, as well as 20000 corrupted APKs. According to the study, the accuracy rate was 94%.

In the study [17], a CNN based model is proposed. Static analysis is performed by using API calls and Opcode sequences as features. In the study using the Drebin dataset, it was stated that the accuracy rate of the proposed fusion model was 97.5%.

The authors of [18] demonstrated the DeepAMD approach by comparing the efficiency of classical machine learning classifiers and deep artificial neural networks with DeepAMD investigations. DeepAMD completed detection achieved the greatest accuracy of 93.4% for malware classification and 92.5% for malware category classification in the static layer. DeepAMD obtained an accuracy of 80.3% for malware classification in the dynamic layer.

Haq et al. [19] presented an Android malware detection framework with permissions, a static analysis technique, and a hybrid DL to detect malware from Android applications. The study was carried out on the Androzoo and AMD datasets. In the proposed study, hybrid DL models and comparative DL-based algorithms were critically evaluated.

Table 1 Comparison of related works

Related Works	Year	Analysis Method	Learning Model	Dataset Names	Performance Results
[11]	2018	Dynamic	RF, KNN, DT	CICAndMal2017	85% (Precision)
[12]	2019	Dynamic	LSTM	CICAndMal2017	97%
[13]	2019	Static & Dynamic	RF	CICAndMal2017 InvesAndMal2019	83.3%
[14]	2019	Dynamic	DT, RF, KNN, SVM, NB	CICAndMal2017	82.80%
[15]	2019	Static	DBN	Google Play and Virus Share	94.64%
[16]	2019	Static	DBN	Google Play	94%
[17]	2019	Static	CNN	Drebin	97.5%
[18]	2020	Static & Dynamic	ANN	AMD	93.4% on Static Layer, 80.3% on Dynamic Layer
[19]	2021	Static	DL	AndroZoo and AMD dataset	99.2%
[20]	2021	Static	MLP, SVM	SEDMDroid	89.07%
[21]	2021	Dynamic	DCGAN_1D-CNN	CICAndMal2017	96.55% (F1- Score)
[22]	2021	Dynamic	ANN	CICAndMal2017	98.4%
[23]	2022	Static	Lightweight CNN	Google Play,Virus Share, AMD	91.27%
[24]	2022	Dynamic	LSTM, NB, RF, SVM, MLP, CNN, GRU, RNN	DroidCollector	95%
[25]	2022	Dynamic	LSTM	CICAndMal2017	99.96%
[33]	2022	Static	NB, SVM	AMD	92.4%

The study [20], introduced the SEDMDroid framework for detecting Android malware. Permission as a static feature in Android malware detection yields an accuracy of 89.07% when API calls and system events are monitored. Additionally, the presented framework extracts datastream information as attributes with a 94.92% accuracy. Based on the test results, the study concludes that the SEMDroid framework is a good way to find malware on Android.

In the proposed study by Luo et al. [21] an encrypted malicious classification method based on the 1D-CNN and DCGAN_1D-CNN model is proposed using the CICAndMal2017 dataset. As a result of experimental studies, it has been reported that DCGAN_1D-CNN model reached 96.55% *F*1 - *Score* value in classification of encrypted malicious traffic.

Authors in [22] performed an Android malware detection system based on neural networks with the CICMalDroid2017 data set, in which different IP encoding methods were used, and achieved an accuracy rate of 98.4%. They presented the IP Address feature, which is one of the features found in the data set they used, in comparison with different IP encoding methods, such as dividing IP into four numbers, converting IP to an integer, and without IP Address

Authors in [23] have proposed a malware detection method they call MAPAS, which can use system resources efficiently and effectively. It analyses the behaviour of malicious applications with API call graphs using CNN deep learning algorithm. CNN was used to explore the common features of the API call graphs of malware, and the lightweight classifier was used as the classification model. In the study, MAPAS and Android malware detection approach called MaMaDroid was compared in terms of memory usage, classification speed and accuracy of classification of unknown malware. The MAPAS method classified applications with 145.8% faster classification, approximately ten times lower memory usage, and 91.27% accuracy. Authors in [24] proposed a paper as compared to the LSTM deep learning model based on the network traffic analysis method of mobile applications with NB, RF, SVM, MLP, CNN, RNN and GRU algorithms. The developed LSTM-based deep learning model has been more successful than the other proposed methods with a 95% accuracy rate. In the study, 10 features of 7845 applications obtained from pcap files of 4704 benign and 3141 malicious applications obtained from the DroidCollector project were used. In addition, the classification models were seen by calculating the feature importance levels of the features used.

Fallah and Bidgoly [25] proposed a method based on the LSTM algorithm for malware detection classification and new and invisible malware families. In the proposed study, the analysis of network traffic data containing dynamic features was carried out on the CICAndMal2017 dataset. In the study, it was detected with an accuracy rate of 99.96% immediately after capturing 50 network traffic flows, and an accuracy rate of 80% was obtained in the detection of new malware.

Yilmaz et al. [33] proposed a study that used machine learning method to classify a data set containing 2854 malicious and 2870 harmless software. It was trained with 116 permission feature (SVM) and Naive Bayes (NB) models of the applications. According to the classification performance results, 90.9% success was obtained from the SVM model and 92.4% from the NB model. In addition, the prediction and learning levels of the models proposed in the study were supported by the ROC curve and AUC values.

A detailed comparison of the studies mentioned in the related works section of the article and the accuracy rates of these studies are presented in Tab. 1. This study is one of the few studies in which static and dynamic analysis techniques are presented together comparatively.

In the study, which is carried out with static and dynamic analysis techniques, different combinations of deep learning methods are presented. The performance rates of the analysis of the three-layer LSTM and the current hybrid method 1D-CNN-LSTM in different detection systems were compared at different solver parameter values of the ANN algorithm and different neuron numbers of the three-layer MLP algorithm. When

the experimental results are compared with the performance ratios presented in Tab. 1, the performance ratio of the experimental studies [13] with the RF algorithm is more promising than the study and the performance results of the ANN and MLP algorithms with different parameter values are higher than the studies [18, 20].

In this study the performance value of the LSTM method is promising in the static analysis technique, and the 1D -CNN-LSTM method in the dynamic analysis technique. According to studies that are open to everyone and use the same data sets, the biggest factor in the success of the experimental study results in this study is undoubtedly the data pre-processing phase. In the experimental results section of the study, the operations performed in the data pre-processing, which we think will provide a perspective to the practitioners, are clearly stated.

When Tab. 1 is examined, the variety of current data sets in which the dynamic analysis technique is applied in the literature is quite low. In this study, it is aimed to contribute to the literature with the high success rates provided by experimental studies using static and dynamic analysis techniques of the same applications.

3 BACKGROUND

To make a malware detection system, some features need to be extracted from Android applications in static and dynamic analysis. Intrusion detection comes automatically after training the system with static and dynamic analysis using learning algorithms. Thus, it provides convenience, efficiency, and speed in the detection of attacks. Therefore, understanding this analysis is important for implementing a robust and efficient detection system.

3.1 Static Analysis

Static analysis is an analysis technique that detects malware without running an application. In the static analysis, the AndroidManifest.xml file is very important. Applications can access some system resources within certain limits. For this purpose, the desired permissions are defined in the AndroidManifest.xml file. In this file, the name of the program, the components of the program, and the necessary permissions for the resources are found [26]. Android application always need the user's explicit permission to store large amounts of data of interest in public memory or to gain access to an unsafe function. It is also extremely important to define these permissions correctly. There are various access permissions that can be given to applications in the application permissions list. For example, there are permissions to access personal data stored on the device (contacts in the phone book, call log, SMS, photos) or internal devices (camera, microphone, phone, GPS receiver) from which personal data can be retrieved. Some applications may want to access other applications' resources or some system resources. Android has developed a permission system for this purpose. An application must declare the permissions it will use by adding <uses- permission <tags to the application manifest. For example, if an application needs to send SMS messages, it should include the line: <uses-permission android: name=" Android. Permission. SEND_SMS"> An application that is granted SMS access can access all SMS messages. These correspondences also include internet banking messages that transmit a one-time code and confirmation transactions [30].

3.2 Dynamic Analysis

Dynamic analysis is also known as behavioural-based analysis. In this technique, a detection is performed that includes information collected during the runtime of the operating system, i.e., during the execution of the program, such as network access and system calls. The capture of network traffic is one of the most important factors for malware analysis [8]. Even if the application does not have internet permission or if the application itself does not generate network traffic, it may miss data or communicate through other applications such as a browser [28, 31].

When the static and dynamic analysis mechanisms are compared, the dynamic analysis mechanism performs better than the static analysis mechanism in detecting attacks using the code hiding technique since the application is analysed at runtime [7, 8, 26, 27].

The static analysis mechanism, on the other hand, is more efficient for detecting previously known attacks. Since static analysis is a passive approach, it is less costly than dynamic analysis, which is an active approach in terms of resources and time since the application is not carried out [27]. The comparison of using both analysis techniques is shown in Fig. 2.

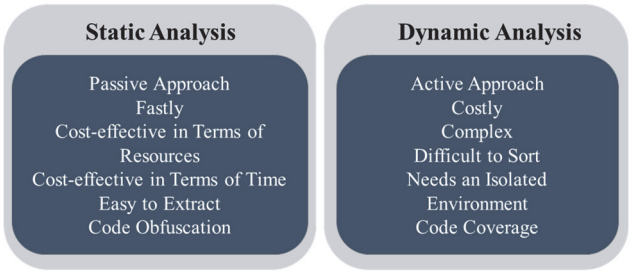

Figure 2 Comparison of dynamic and static analysis techniques

3.3 Machine Learning Models

Machine learning is an algorithm system that allows software programs to make more accurate predictions from data by optimizing algorithms without explicit programming. The mainstay of these algorithms is to generate new output data by making predictions from the input data, instead of following static processing steps, and to work by creating a model by updating these outputs [9].

Random Forest (RF) algorithms are algorithms that are used in classification and regression problems with a high prediction rate and generate ensembles with a randomly selected set of sub-trees by creating many prediction models. RF learning model is a method used on categorical and continuous data in all data sets without any size problem [30, 34].

Decision Tree (DT) is an algorithm that creates a tree structure model consisting of decision nodes and leaf nodes according to the feature and target that can be used in classification and regression problems. It is based on the rule of recursively dividing the input data into groups with the help of a clustering algorithm. The clustering process

continues in depth until all elements of the group have the same class label. It is possible to see the decisions that affect the result in this algorithm. For this reason, it is a popular solution used in threat detection studies [29, 35].

Multilayer Perceptron (MLP) has a structure in which many neurons with non-linear activation functions in architectural terms are hierarchically connected to each other. It uses a learning system called back-propagation [10].

Artificial Neural Network (ANN) is a learning event that is usually organized in layers and occurs when neurons in each layer connect with neurons in the previous and next layer. They are the systems where learning takes place at the end of the training process by processing the information received from the neurons in the intermediate layers from the input layer to the output layer [18, 36].

MLP and ANN algorithms are like each other, but MLP is a type of the ANN algorithm. MLP is a structure consisting of at least three layers, one of which is the hidden layer and in that there can be one or more non-linear layers. The ANN model can be one neuron and adding more neurons until the network performance in estimating the output is satisfactory to create the best ANN modelled with the least number of neurons.

Long short-term memory (LSTM) algorithm is a variant of the Recursive Neural Network (RNN), which does not provide effective results in training problems due to its short-term memory and occurs by eliminating these problems. LSTM is widely used in sequential or time-series problems because it can learn long-term dependencies with its memory-transitive mechanism. The basic LSTM architecture consists of input, output, forget gates and memory neurons [12].

Convolutional Neural Network (CNN) has many applications such as image classification, detection, object analysis. CNN can obtain local features from the layer inputs and add them to the lower layers. CNN consists of convolution, pooling and fully layers. In this study, we combined one- dimensional CNN (1D-CNN) with LSTM network to detect malware by reducing the feature size of feature detecting 1D-CNN [21].

3.4 Datasets

CICInvesAndMal2019 [13] is the follow-up data set to CICAndMal2017 [11] in which benign and malware Android applications are evaluated on smart devices. On smart devices, benign and malware Android applications have been evaluated and produced. The CICAndMal2017 data set, which contains 426 malware and 5065 benign samples developed by merging innocuous samples from Google Play with malware samples from a variety of sources, was published in 2018. Continuous features in the CICAndMal2017 data set include logs, network traffic, and API requests, while discrete characteristics include battery use, permissions, network traffic, and memory dumps. CICInvesAndMal2019 covers static features such as permissions and intents, as well as dynamic elements such as API calls and all generated log files (80 network- flows). The InvesAndMal2019 data set contains 426 malware and 5065 benign labeled samples grouped into four categories. These data sets contain four distinct types of malware: adware, ransomware, scareware, and SMS malware. Information about the data set used in the study is shown in Tab. 2.

Table 2 Details of dataset

	Dataset and features		
	Features	CIC-And Mal2017	CIC-Inves AndMal 2019
	Year	2018	2019
	#Benign	5065	
	#Malware	426	
	#Feature	84	8115
Captured static features	Permission	×	✓
	Intent	×	✓
	State	×	✓
	Cert.	×	×
	Source Code	×	×
Captured dynamic features	API Call	✓	✓
	Newt.	✓	✓
	Sys. Call	×	✓
	Log	×	✓

4 PROPOSED METHOD

In this study, it is aimed to present deep learning-based malware binary classification comparatively by using static and dynamic analysis techniques. In this study, traditional machine learning algorithms are also shown, along with how well they work in different analysis techniques.

First, in static analysis binary classification, samples are classified as malware or benign in the data set containing permissions and intents with test and training samples. Then, malware in four different categories in malware dynamic analysis classification was combined to create an up-to-date data set for binary classification. By preprocessing these up-to-date datasets, a comparative analysis with traditional machine learning algorithms DT, RF and LSTM, CNN-LSTM, ANN and MLP deep learning algorithms is presented.

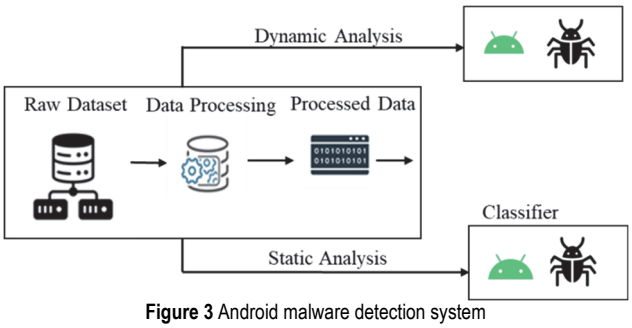

Figure 3 Android malware detection system

The process and design of the proposed Android malware detection approach are shown in Fig. 3. Some factors were effective in the selection of the methods used. The factor in the selection of the RF classifier is that it gives effective results, especially in data sets with uneven distribution. The fact that the data set using static analysis features is unbalanced in terms of the number of benign and malware features has been effective in the use of this method. DT classifiers are fast to train and test. It has a structure that uses data sets with large sample numbers by dividing them into small sample groups.

The data sets used in this study differed in terms of the number of samples, which was effective in choosing this method. Many activation functions can be used in the ANN architecture. It has a structure suitable for development, with different network structures. For example, the sigmoid function is frequently used for classification. An ANN can solve a problem using a single neuron. The parameter values used in different methods of the ANN algorithm in static and dynamic analysis techniques are shown in Tab. 3. Classification ability is possible by increasing the number of layers. In this study, a three-layer MLP structure using different activation functions was used.

Table 3 ANN classifier parameters

ANN Parameters	ANN-1	ANN-2	ANN-3
Solver	adam	sgd	lbfgs
Num. of neurons	64		
Max. iter	150		
Activ. function	relu		
Learning rate in it	0.2		-
Alpha	$1e^{-5}$		

The LSTM has three gates that control and update neurons: the forget gate, the input gate, and the output gate.

The forget gate controls what information in neurons will be forgotten based on the input data. LSTM has been a preferred architecture in terms of keeping the inputs in long-term memory and being a solution to the vanishing gradient problem. By their nature, gates use hyperbolic tangent and sigmoid activation functions. The parameter values used in the LSTM model, which was created using a three-layer structure in the static and dynamic analysis technique, are shown in Tab. 4.

Table 4 LSTM classifier parameters

Parameter values	Layers
lstm_1	unit = 128 activation = relu return_sequences = true
dropout_1	0.2
lstm_2	unit = 64 activation = relu return_sequences = true
dropout_2	0.3
lstm_3	unit = 16 activation = relu return_sequences = false
dropout_3	0.3
dense	unit = 1 activation = sigmoid

1D-CNNs are very suitable for use in mobile devices, especially with low energy and processing power, due to their low computational cost and no special hardware requirements [21, 23]. In the 1D-CNN-LSTM algorithm, first a deep model was created as the input layer, convolution layer, and pool layer, and the reconstructed features that reduced the feature size are input into the LSTM algorithm, and a detection model was created in the malware classification. The architecture of the proposed hybrid 1D-CNN -LSTM classification model is presented in Fig. 4.

The ability of LSTM to store in memory during time steps and to cascade sequentially connected sequences is combined with the 1D-CNN algorithm to determine the CNN-LSTM model with static and dynamic analysis techniques. The parameter values used in the 1D-CNN-LSTM classification model are shown in Tab. 5.

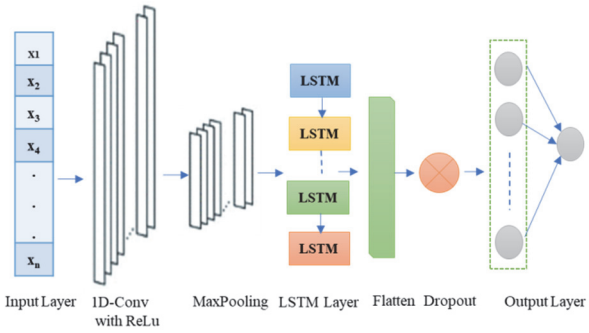

Figure 4 Architecture of the hybrid 1D-CNN -LSTM

Table 5 1D-CNN-LSTM classifier parameters

Parameter values	Layer values
conv1d_1	unit = 128 kernel_size = 3 activation = relu
max_pooling1d_1_	pool_size = 2
conv1d_2	unit = 64 kernel_size = 3 activation = relu
max_pooling1d_2	pool_size = 2
lstm	unit = 8
dropuout_1	0.4
dense_1	unit = 1 activation = sigmoid

The confusion matrix was run for the accuracy of classifiers, and *F*1 - *Score* were evaluated. *Precision* or *Recall* were used to measure *Accuracy* and *F*1 - *Score* was used for the imbalanced data. The formulas of the performance evaluation metrics used are given below [32]. *TP*: Predicted Positive, Actual Positive; *TN*: Predicted Negative, Actual Negative; *FP*: Predicted Positive, Actual Negative; *FN*: Predicted Negative, Actual Positive.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1\text{-}Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

Table 6 System properties

Property	Value
Processor	i7-8th Gen(8700K)
Number of Core	6
Number of Threads	12
Turbo Boost	4.70 GHz
Cache L1/L2/L3	64K/256K/12MB
Memory Type	DDR4- 2666
Memory Size	16 GB
Operating System	Windows-10, 64-bit
Graphics Card	Nvidia G-Sync

Python 3.8.3 was used to obtain the experimental performance results, in which classification was made according to the analysis methods, and the features of the computing platform are depicted in Tab. 6.

5 EXPERIMENTAL RESULTS

In this section, comparative performance results are presented using static and dynamic features using machine learning algorithms.

5.1 Malware Binary Detection on Static Analysis

The system consists of two main elements: pre-processing and classification. In the data set containing static features, two separate files used for testing and training are kept in CSV format. To obtain the best performance in machine learning algorithms in the pre-processing of the data set, which includes the permissions and intentions of the applications, NaN (Not-a-Number) and duplication removal processes are applied first using the NumPy library. In this study, the family and category columns in the data set containing static properties were removed, since binary classification was performed. A MinMax scaling process was applied for feature normalization in the selected data set. Normalization refers to the re-scaling of real-valued numeric attributes to a fixed range (e.g., 0 and 1). There are 8115 features in the data set, which has 60% training data samples and 40% test data samples. Static analysis binary classification performance results using learning methods are presented in Tab. 7.

Considering the results of traditional machine learning algorithms according to the performance results, it is seen that the RF classifier provides 95.27% accuracy.

Table 7 Static analysis binary malware classification performance

	Accuracy / %	*F*1 - *Score* / %	*Recall* / %	*Precision* / %
RF	95.27	97.28	98.34	96.25
DT	91.64	94.57	92.67	96.56
ANN-1	92.26	95.19	92.36	98.23
ANN-2	94.16	96.18	96.19	97.23
ANN-3	93.33	95.89	95.42	96.38
MLP-1	98.41	98.11	98.65	97.59
MLP-2	94.50	96.16	95.24	97.13
MLP-3	95.26	94.86	95.26	94.31
LSTM	98.75	97.35	97.03	97.69
CNN-LSTM	98.02	98.87	98.31	99.44

The RF algorithm randomly selects different subsets from both the data set and the feature set, trains them, and classifies them according to the most votes among the predictions of the decision trees it creates.

When the ANN results are examined, it is shown that the classifier whose parameter values are specified in Tab. 3, whose highest accuracy rate is 94.16%, is called ANN-2. One of the solver parameters, which is one of the parameter values used as the weight optimization value, "adam" works better than the "lbfgs" parameter value in large data sets. When the result of the experimental study is examined, it is seen that the model called ANN-3 gives better results than the ANN-1 model. In addition, the initial learning rate value is used only when the solver parameter value is only "sgd" or "adam", so it is not used in the "lbfgs" parameter.

When the MLP results are examined, it is shown that the classifier with the highest accuracy rate is 98.12%, that model is called MLP-1 and has the parameter values specified in Tab. 8.

Table 8 Static analysis of MLP classifier parameters

	MLP-1	MLP-2	MLP-3
Input Layer	8111	8111	8111
1st Layer	256	256	256
2nd Layer	128	64	64
3rd Layer	64	256	256
Output Layer	1	1	1
Activation Func. of Layer	Relu	Relu	Logistic
Output Activation Function	Sigmoid		
Classifier Opt.	Adam		
Epoch	150		
Dropout	0.5		
Loss Func.	Binary Cross Entropy		
Batch Size	32		

In both studies using static and dynamic analysis techniques of LSTM and 1D -CNN-LSTM methods, the loss function "binary_cross entropy" was used, and the optimization method "adam" hyper parameters since they belong to one of the candidate solutions in binary classification. When the performance results are examined, it is seen that the LSTM algorithm has reached the highest accuracy value with 98.75% performance in the static analysis classification against all learning algorithms.

5.2 Malware Binary Detection on Dynamic Analysis

In the pre-processing stage of data collection with dynamic features, the Nan and duplicative removal processes were used first, followed by the rest of the processing. In the data set used for attack detection and identification over network traffic, there are four malware categories and the families that correspond to each category are included in the data set. Because of this, all malware instances are labelled as 1 and all benign examples as 0, and the Label Map function is used to classify all categories of malware, including Ransomware, Adware, SMS Malware, Scareware, and SMS Trojans as a single binary classification. Regarding the number of malwares, the data set used is benign and unstable in terms of the number of malwares.

As a result, a merging process was carried out in such a way that the data set used could be balanced. By randomly picking samples from benign software, we built an up-to-date data set of 559150 rows containing 281076 malware and 278074 benign samples. There are two types of IP addresses in the generated data set: source IP addresses and destination IP addresses. IP address information is a consideration in network attacks that compromise system performance [22]. IP addresses are used after being converted to integer format using theipaddress" function. The timestamp is another characteristic that must be translated to a format suitable for machine learning classification on the data set. It has been transformed to the "str" data type in this feature based

on its frequency of occurrence within the date range. Following all pre-processing steps, the classification procedure utilized 71 characteristics, including dynamic features. Dynamic binary classification performance results using machine learning methods are presented in Tab. 9. According to the performance results, the highest accuracy rate of 95.26% was calculated with the 1D-CNN-LSTM classifier.

Table 9 Dynamic analysis binary malware classification performance

	Accuracy / %	*F*1 - *Score* / %	*Recall* / %	*Precision* / %
RF	92.73	92.66	92.63	93.56
DT	90.09	89.99	86.23	94.12
ANN-1	86.05	86.10	85.13	87.06
ANN-2	85.35	86.37	85.23	87.56
ANN-3	86.07	86.68	86.27	87.13
MLP-1	94.64	95.42	99.57	91.63
MLP-2	93.78	94.55	98.69	90.75
MLP-3	94.25	94.66	98.33	91.28
LSTM	94.52	95.18	98.11	92.43
CNN-LSTM	95.26	95.57	99.26	92.15

According to the classification results using the parameter values of the ANN classifier, it has been observed that the high number of static analysis features is among the reasons that increase the performance rate. According to the ANN performance results, ANN-1 and ANN-3 have the same accuracy rate, and the highest accuracy rate was 86.07%.

When the MLP results are examined, it is shown that the classifiers with the highest accuracy rate 94.64%, called MLP-1 which have the parameter values specified in Tab. 10.

Table 10 Dynamic analysis MLP classifier parameters

	MLP-1	MLP-2	MLP-3
Input Layer	71	71	71
1st Layer	32	64	64
2nd Layer	64	32	32
3rd Layer	32	64	16
Output Layer	1	1	1
Activation Func. of Layer	Relu		
Output Activation Function	Sigmoid		
Classifier Opt.	Adam		
Epoch	150		
Dropout	0.5		
Loss Function	Binary Cross Entropy		
Batch Size	32		

According to the dynamic analysis classification results obtained, it is clearly seen that the 1D-CNN-LSTM classifier has achieved high performance in these techniques. In this study, we performed ten experiments with two different analysis methods. We examined the effectiveness of machine learning algorithms in detecting mobile Android malware with static and dynamic features on the CIC-AndMal2017 and the second part, CIC-InvesAndMal2019 data sets. According to the results, the accuracy value deep learning classification algorithm showed high performance in all scenarios among the proposed methods in both analysis methods. We observed that static analysis experimental studies had a higher performance rate on average than dynamic analysis. Our findings show that the values in the data set containing dynamic features do not show normal distribution, especially due to the network traffic data, have fewer features, and thus the performance ratios in classification are lower. The proposed algorithms are adequate for detecting a significant amount of malware.

6 CONCLUSION

With the continued growth of mobile devices and applications in recent years, cyber security has gained increased attention. Android, as the most common operating system for mobile devices, has been the primary target of attackers looking to harm or exploit these devices to obtain end users' financial or personal information. To access these devices, intruders aim to upload some malware to the target machines. As a result, security researchers concentrated on detecting these malwares before it activated or caused harm to mobile devices. Deep learning is a useful method for self-learning from previous experiences and then applying that learning without requiring human intervention. In this paper, we proposed a deep learning-based malware detection system by using different approaches and depicting the results in a comparative way. The static and dynamic analysis of this suspicious software is detailed by giving their experimental results, which show the effectiveness of the proposed approach. When the results of comparative experimental studies are examined, it has been revealed that LSTM with an accuracy rate of 98.75% in static analysis classification and CNN-LSTM deep learning algorithms with an accuracy rate of 95.26% in dynamic analysis classification has the highest performance in studies where static analysis and dynamic analysis features are evaluated. Analysis features, such as the number of layers and neurons, and the effect of parameter values on the success rate are briefly shown. As a future work, it is aimed to analyse the time sequence effects of the observed behaviours in a dynamic analysis concept with the use of some deep learning approaches.

Acknowledgements

This work has been supported by Marmara University Scientific Research Projects Coordination Unit under grant number FDK-2020-10066.

7 REFERENCES

[1] Stat Counter Global Stats. (2022, June 26). Desktop vs Mobile vs Tablet Market Share Worldwide 2009 to 2022. Retrieved from https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet.

[2] Stat Counter Global Stats. (2022, Feb. 26). Mobile Operating System Market Share Worldwide from January 2012 to January 2022.
Retrieved from https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet

[3] Zimperium. (2022, April 21). 2022 Global Mobile Threat Report. Zimperium.

Retrieved from https://www.zimperium.com/global-mobile-threat-report/.

[4] Kim, J. & Lee, S. (2021). Malicious Behavior Detection Method Using API Sequence in Binary Execution Path. *Tehnički Vjesnik - Technical Gazette*, *28*(3), 810-818. https://doi.org/10.17559/TV-20210202132203

[5] McGraw, G. & Morrisett, G. (2000). Attacking malicious code: A report to the infosec research council. *IEEE Software*, *17*(5), 33-41. https://doi.org/10.1109/52.877857

[6] Tahir, R. (2018). A study on malware and malware detection techniques. *International Journal of Education and Management Engineering*, *8*(2), 20. https://doi.org/10.5815/ijeme.2018.02.03.

[7] Yu, B., Fang, Y., Yang, Q., Tang, Y., & Liu, L. (2018). A survey of malware behavior description and analysis. *Frontiers of Information Technology & Electronic Engineering*, *19*(5), 583-603. https://doi.org/10.1631/FITEE.1601745

[8] Sihwail, R., Omar, K., & Ariffin, K. Z. (2018). A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. *International Journal on Advanced Science, Engineering and Information Technology*, *8*(4-2), 1662-1671. https://doi.org/10.18517/ijaseit.8.4-2.6827

[9] Bayazit, E. C., Sahingoz, O. K., & Dogan, B. (2020, June). Malware detection in Android systems with traditional machine learning models: a survey. *International Congress on Human-Computer Interaction, Optimization, and Robotic Applications, (HORA2020)*, 1-8. https://doi.org/10.1109/HORA49412.2020.9152840

[10] Bayazit, E. C., Sahingoz, O. K., & Dogan, B. (2022, June). A Deep Learning Based Android Malware Detection System with Static Analysis. *International Congress on Human-Computer Interaction, Optimization, And Robotic Applications (HORA2022)*, 1-6. https://doi.org/10.1109/HORA55278.2022.9800057

[11] Lashkari, A. H., Kadir, A. F. A., Taheri, L., & Ghorbani, A. A. (2018, October). Toward developing a systematic approach to generate benchmark Android malware datasets and classification. *International Carnahan Conference on Security Technology (ICCST)*, 1-7. https://doi.org/10.1109/CCST.2018.8585560

[12] Bibi, I., Akhunzada, A., Malik, J., Ahmed, G., & Raza, M. (2019, August). An effective Android ransomware detection through multi-factor feature filtration and recurrent neural network.*UK/China emerging technologies (UCET)*, 1-4. https://doi.org/10.1109/UCET.2019.8881884

[13] Taheri, L., Kadir, A. F. A., & Lashkari, A. H. (2019, October). Extensible android malware detection and family classification using network-flows and API-calls. *International Carnahan Conference on Security Technology (ICCST)*,1-8. https://doi.org/10.1109/CCST.2019.8888430

[14] Noorbehbahani, F., Rasouli, F., & Saberi, M. (2019, August). Analysis of machine learning techniques for ransomware detection. *16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, 128-133. https://doi.org/10.1109/ISCISC48546.2019.8985139

[15] Sandeep, H. R. (2019, May). Static analysis of Android malware detection using deep learning. *International Conference on Intelligent Computing and Control Systems (ICCS)*, 841-845. https://doi.org/10.1109/ICCS45141.2019.9065765

[16] Mahindru, A. & Sangal, A. L. (2019, October). Deepdroid: feature selection approach to detect android malware using deep learning.*10th International Conference on Software Engineering and Service Science (ICSESS)*, 16-19. https://doi.org/10.1109/ICSESS47205.2019.9040821

[17] Ding, Y., Hu, J., Xu, W., & Zhang, X. (2019, July). A deep feature fusion method for Android malware detection. In *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*, 1-6. https://doi.org/10.1109/ICMLC48188.2019.8949298

[18] Imtiaz, S. I., ur Rehman, S., Javed, A. R., Jalil, Z., Liu, X., & Alnumay, W. S. (2021). DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network. *Future Generation Computer Systems*, *115*, 844-856. https://doi.org/10.1016/j.future.2020.10.008

[19] Haq, I. U., Khan, T. A., & Akhunzada, A. (2021). A dynamic robust DL-based model for android malware detection. *IEEE Access*, *9*, 74510-74521. https://doi.org/10.1109/ACCESS.2021.3079370

[20] Zhu, H., Li, Y., Li, R., Li, J., You, Z., & Song, H. (2020). SEDMDroid: An enhanced stacking ensemble framework for Android malware detection. *IEEE Transactions on Network Science and Engineering*, *8*(2), 984-994. https://doi.org/10.1109/TNSE.2020.2996379

[21] Luo, W., Liu, Z., Zhao, R., Chen, J., & Deng, X. (2021, December). Malicious HTTPS Traffic Classification Algorithm Based on DCGAN_1D-CNN. *IEEE Conference on Telecommunications, Optics and Computer Science (TOCS),*20-25. https://doi.org/10.1109/TOCS53301.2021.9688753

[22] Bayazit, E. C., Sahingoz, O. K., & Dogan, B. (2021, June). Neural networkbased Android malware detection with different IP coding methods. *International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA 2021)*, 1-6. https://doi.org/10.1109/HORA52670.2021.9461302

[23] Kim, J., Ban, Y., Ko, E., Cho, H., & Yi, J. H. (2022). MAPAS: a practical deep learning-based android malware detection system. *International Journal of Information Security*, 1-14. https://doi.org/10.1007/s10207-022-00579-6

[24] Anıl, U. T. K. U. (2022). Ağ trafiği analizi ile derin öğrenme tabanlı Android kötücül yazılım tespiti. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, *37*(4), 1823-1838. https://doi.org/10.17341/gazimmfd.937374

[25] Fallah, S. & Bidgoly, A. J. (2022). Android malware detection using network traffic based on sequential deep learning models. *Software: Practice and Experience*, *52*(9), 1987-2004. https://doi.org/10.1002/spe.3112

[26] Wu, Q., Zhu, X., & Liu, B. (2021). A survey of Android malware static detection technology based on machine learning. *Mobile Information Systems*, *2021*, 1-18. https://doi.org/10.1155/2021/8896013

[27] Lee, S., Jeon, H., & Park, G. (2021). Design of Automation Environment for Analyzing Various IoT Malware. *Tehnički Vjesnik - Technical Gazette*, *28*(3), 827-835. https://doi.org/10.17559/TV-20210202131602

[28] Premkumar, M., Sundararajan, T. V. P., & Mohanbabu, G. (2022). Dynamic Defense Mechanism for DoS Attacks in Wireless Environments Using Hybrid Intrusion Detection System and Statistical Approaches. *Tehnički Vjesnik - Technical Gazette*, *29*(3), 965-970. https://doi.org/10.17559/TV-20210604113859

[29] Quinlan, J. R. (1986). Induction of decision trees. *Mach Learn*, *1*, 81-106. https://doi.org/10.1007/BF00116251

[30] Kim, H., Cho, T., Ahn, G. J., & Hyun Yi, J. (2018). Risk assessment of mobile applications based on machine learned malware dataset. *Multimedia Tools and Applications*, *77*(4), 5027-5042. https://doi.org/10.1007/s11042-017-4756-0

[31] Cafuta, D., Sruk, V., & Dodig, I. (2018). Fast-flux botnet detection based on traffic response and search engines credit worthiness. *Tehnički Vjesnik - Technical Gazette*, *25*(2), 390-400. https://doi.org/10.17559/TV-20161012115204

[32] Kohavi, R. & John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, *97*(1-2), 273-324. https://doi.org/10.1016/S0004-3702(97)00043-X

[33] Yılmaz, A. B., Taspınar, Y. S., & Koklu, M. (2022). Classification of Malicious Android Applications Using Naive Bayes and Support Vector Machine Algorithms. International Journal of Intelligent Systems and Applications in Engineering, *10*(2), 269-274.

[34] Al-doorı, S. K. S., Taspınar, Y. S. & Koklu, M. (2021). Distracted Driving Detection with Machine Learning Methods by CNN Based Feature Extraction. *International Journal of Applied Mathematics Electronics and Computers*, *9*(4), 116-121. https://doi.org/10.18100/ijamec.1035749

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

[36] Taspinar, Y. S., Cinar, I., & Koklu, M. (2022). Classification by a stacking model using CNN features for COVID-19 infection diagnosis. *Journal of X-ray science and technology*, 1-16. https://doi.org/10.3233/XST-211031

Contact information:

Esra Calik BAYAZIT
(Corresponding author)
1) Computer Engineering Department,
Fatih Sultan Mehmet Vakif University, Beyoglu, Istanbul, 34445, Turkey
2) Marmara University Institute of Science
E-mail: ecalik@fsm.edu.tr

Ozgur Koray SAHINGOZ
Computer Engineering Department,
Biruni University, Topkapi, Istanbul, 34093, Turkey
E-mail: osahingoz@biruni.edu.tr

Buket DOGAN
Department of Computer Engineering, Faculty of Technology,
Marmara University, Basibuyuk, Istanbul, 34854, Turkey
E-mail: buketb@marmara.edu.tr