Journal of Theoretical and Applied Information Technology
15th January 2024. Vol.102. No 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CYBERSECURITY: MALWARE MULTI-ATTACK DETECTOR ON ANDROID-BASED DEVICES USING DEEP LEARNING METHODS

MUSTAFA ABABNEH[1] AND AAYAT ALJARRAH[2]

[1]Department of Computer Science, Kingdom of Bahrain – Riffa Building 287, Road 3903, Block 939, P.O. Box 40434 Bahrain

[2]Department of Computer Science, Nasser Vocational Training Centre Jau, Road No. 5712, Jaww P.O. Box 80240, Bahrain

Email: mustafa.ababneh86@gmail.com, aayat.aljarrah@nvtc.edu.bh

ABSTRACT

Android-based devices are currently a prime target for cyber-attackers. New malware is being developed and released, with devastating effects on sensitive information lost and ransom payments. Android developers and users continue to look for holistic methods of detecting all types of malware instead of individual ones. The aim of this study is to test the combined impact of deep learning (DL) methods on detecting malware with multi-attack features on Android devices. A malware multi-attack detector (MMAD) combined DL methods: deep neural networks (DNN), recurrent neural networks (RNN), convolutional neural networks (CNN), multilayer perceptron's (MLP), and end-to-end (E2E). Each of these methods detects specific types of malware. Different types of malware attacks, including benign ones, were used to train and test the MMAD model. Experimental results indicated that the proposed MMAD model was efficient in detecting eleven types of malware attacks with a high and constant multi-classification capability. Our results with 96.54% accuracy, 95.38% precision, 92.65% recall, and a 94.66 F-score showed that the MMAD approach is effective, efficient, and simple to use.

Keywords: *Android Devices; Malware Multi-Attack Detectors; Deep Learning Methods*.

1. INTRODUCTION

Cybersecurity attacks are malicious acts carried out by hackers, cyberpunks, crackers, keyloggers, and hacktivists with the dangerous intent of stealing data, breaching, or interrupting computing systems (De Arroyabe et al., 2023) [23]. These cybercriminals unleashed attacks using single or multiple approaches against a target device (Shaikh & Siponen, 2023) [54]. A malicious cyber-attack may steal sensitive information, disable devices, or use a compromised computer as a launching pad. The attacks include malware, denial of service, man in the middle, and injection, which can be difficult to detect through simple methods (Florackis et al., 2023) [27].

The most prevalent form of cyberattack is malware. Malware is any harmful software, including viruses, spyware, trojan horses, worms, backdoors, and ransomware (Mijwil et al., 2023) [42]. With mobile devices becoming more and more important, cybercriminals are concentrating more on them. As a result, the range of cyber-attacks aimed at these devices has increased nowadays (Aslan et al., 2023) [12]. There are various ways to target Android devices. This covers potential malicious software, network-level intrusions, and the use of mobile OS and device susceptibilities of the mother systems (Ozdamli et al., 2022 [50] , Yan et al., 2022; Aslan et al., 2023) [12]. New malware is being discovered every day. Applications for Android are frequently used to handle sensitive data, making them major targets of malware attacks.

The increased level of information sharing and open interface exposure between ICTs and the public has increased malicious assaults in both intra-terminal and inter-terminal networks. Service interruptions, information outflows, and other problems could result from such attacks. Attacks with multiple stages are carried out progressively in several steps. Each stage of the attack results in the attackers learning more about the target system, enabling the next stage of the attack (Hu et al., 2022) [22]. In the beginning, the attacker tries to gather data and take advantage of the target system's vulnerabilities.

Subsequently, the attacker tries to sneak inside the target system and compromise its resources using multi-attack accessibility (Li et al., 2023) [75].

Fortifying the Android operating system against such cyberattacks is possible using a variety of malware detection techniques. These malware detection approaches mostly use features contained in Android applications (Bhat & Dutta, 2022) [16]. The Android defence system is adversely affected when the analytics data keeps expanding (Wang et al., 2020) [65]. Multi-approach detection with improved accuracy considering large data is required.

Deep learning is currently used in cybersecurity to protect user privacy, recognize unusual malware, and complicate numerous attacks (Xin et al., 2018) [69]. It rapidly detects malware in files and keeps track of malware risks (Sarker, 2021) [53]. In addition to deep learning functions for cybersecurity, deep learning algorithms are better at detecting and classifying attacks because they are independent of any well-known risk patterns. As a result, it can perform better than other computing methods in terms of accuracy and computing time. Deep learning methods can effectively extract the data features and apply them to detect malware from cyber attackers (Tuor et al., 2017 [61] ; Mijwil et al., 2023) [42]. It has been suggested that the combination of these deep learning methods can be robust and resilient against multiple malware detections, particularly from attackers using malware with multiple pathways.

High-potential malware can be difficult to spot for two reasons (Akhtar & Feng, 2022) [3]. The first is that it can be challenging to tell whether software is harmful. The second is that malware employs technical techniques to make it more challenging to detect. Antivirus software reportedly fails to identify 35% of malware (Islam et al., 2023) [32].

1.1. Problem Statement

There has not been reported information on the combination of deep learning methods to detect malware with multiple attacks that are difficult to discover. Adware, bots, fileless malware, keyloggers, mobile malware, ransomware, rootkits, spyware, trojans, wiper malware, and worms are difficult to detect due to their evasiveness (Tahir et al., 2018) [59]. Malware can change into various versions or use multiple pathways, which decreases the likelihood that they will be detected owing to the variances in their signatures (Pai et al., 2022) [47]. Some malware executes code that fingerprints the environment to avoid detection, while others can confuse automated tools' detection algorithms due to their multi-attack features (Du et al., 2022) [16]. Malware can do this by switching the server it uses to avoid being discovered by technology. This occurs when malware only operates at specific moments or in response to specific user actions, allowing it to operate at susceptible times like the boot process while dormant the rest of the time, masking internal data so that malware cannot be found by automated methods (Aldhyani & Alkahtani, 2022) [8].

information-hiding strategies, such as spyware and fileless malware, which runs in memory rather than using files and makes use of already-existing system tools to commit crimes (Kaushik et al., 2023) [36], while ransomware can spread via the network and files, affecting both software and hardware (Zhu et al., 2023) [75]. These attacks have increased in frequency recently, accounting for 33% of malware attacks in 2023, while 493.33 million ransomware attacks were reported in 2022 (Neprash et al., 2022 [45] ; Dameff et al., 2023) [22]. Over 70% of these attacks are carried out by cyber-attackers using multi-attack strategies that are difficult to detect (Dameff et al., 2023) [22]. Even though these attacks are difficult to carry out, they are becoming more common; therefore, a robust multi-faceted approach is required to tackle this problem.

The aim of this study is to test the combined impact of deep learning methods on detecting malware with multi-attack features on Android devices. To achieve this, a detection model combining deep learning would need to be created to detect attacks when the Android system is subject to multiple intrusions.

1.2. Contribution

The contribution of this work includes:

i. We have proposed combined deep learning methods, including DNN, RNN, CNN, MLP, and E2E models, in a structured manner for malware detection.
ii. Each model detects some specific types of malware, followed by aggregate detection for a holistic analysis of the multiple attacks.
iii. The model was trained and tested using datasets including multi malicious attack types instead of single dataset, making it capable of recognizing any combination of attacks and categorizing each clearly.
iv. Holistic detection of malware using holistic deep learning approaches

2. RELATED WORKS

2.1. Cybersecurity

Cyber-attacks and data breaches can affect any

network and Android devices. Attacks can potentially completely damage the Android system, and sensitive information can be lost (Abdel Ouahab et al., 2022) [1][2]. Information theft, monetary gain, espionage, or sabotage are a few examples of the attacker's motivations (Mijwil et al., 2023) [42].

Bhandari et al. (2023) [14] used distributed deep neural network-based middleware for cyber-attacks detection in the smart IoT ecosystem. The deep neural network (DNN) model used in IoT devices is subjected to performance and concurrency testing in order to confirm the viability of in-production deployment. On each of the used datasets, the ML models were able to show nearly 93 percent detection accuracy and a 92 percent f1-score. The models' output demonstrates that the system properly and effectively detects malware and attacks in smart environments.

An Android malware detection model called RHSODL-AMD is presented by Albakri et al. (2023) [7]. The method described comprises identifying the Application Programming Interface (API) calls and the most important permissions, which produces effective differentiation between legitimate software and malicious software. As a result, a strategy called RHSO-FS was developed to enhance the classification outcomes. The Adamax optimizer with attention recurrent autoencoder (ARAE) model is also used to identify malware on Android. The RHSODL-AMD technique's experimental validation on the Andro-AutoPsy dataset demonstrates its promising performance, with a maximum accuracy of 99.05 percent.

The Optimal Ensemble Learning Approach for Cybersecurity (AAMD-OELAC) method is presented by Alamro et al. (2023) [5]. The automatic classification and identification of Android malware is the primary goal of the AAMD-OELAC method. The AAMD-OELAC technique uses three machine learning (ML) models—the kernel extreme learning machine (KELM), regularised random vector functional link neural network, and least square support vector machine—for the identification of Android malware (RRVFLN). Last, the hunter-prey optimization (HPO) method is used to tune the three DL models' parameters for better malware detection outcomes. The simulation results demonstrated how AAMD-OELAC technology outperforms other currently employed techniques.

Abdel Ouahab et al. (2022) [1][2], proposed an intelligent cybersecurity framework specialized in malware attacks in a layered architecture. The framework core layer processes unknown datasets of harmful software after receiving the unknown malware and using the malware visualisation approach. The algorithms K-Nearest Neighbor, Decision Tree, and Random Forest are used to group malware data into families. The suggested Intelligent Cybersecurity Framework is implemented in an intuitive graphic user interface. The random forest method performs the classification task with the highest level of precision (97.6%).

2.2. Malware Attacks on Android

The most frequent type of threat to Android is malware. Android malware is defined as malware that particularly targets the Android operating system and damages or steals data from mobile devices running Android (Bhat & Dutta, 2022 [16] ; Albakri et al., 2023) [7]. Malware can also sneakily install itself without the user's knowledge or agreement by exploiting flaws in operating systems or web browsers (Da Costa & Moia, 2012). Once installed, malware has the ability to track user activity, communicate sensitive information to the attacker, help the perpetrator breach other network targets, and even make the user's device a member of a botnet that the attacker uses for nefarious purposes (Varlioglu et al., 2022) [62]. Trojans, spyware, adware, ransomware, worms, botnets, and backdoors are the classifications given to these malware (Gibert et al., 2020; Mahdavifar et al., 2020) [39]. Malware is typically categorised by researchers into sub-types, as shown in Table 1.

Table 1: categorisation of malwares and their target effect

Type	Target Effect	Real-World example	Reference
Ransomware	Prevents the victim from accessing data until the ransom is paid	RYUK	Dameff et al. (2023)[22]
Fileless Malware	Alters native OS files	Astaroth	Varlioglu et al. (2022) [62]

Spyware	Gathers information on user task without the user's awareness	DarkHotel	Qabalin et al. (2022) [49]
Adware	Operates undesirable ads	Fireball	Alani et al. (2022) [6]
Trojans	Masquerades itself as wanted code	Emotet	Kanaker et al. (2022) [33]
Worms	Replicates itself to spread within a network	Stuxnet	Pan et al. (2022) [48]
Rootkits	Provides hackers access to a victim's device remotely.	Zacinlo	Mohammadzad & Karimpour (2023) [44]
Keyloggers	Observes user's keystrokes	Olympic Vision	Bhardwaj & Goundar (2020) [15]
Bots	Unleashes a massive barrage of attacks	Echobot	Alahmadi et al. (2020) [4]
MobileMalware	Infects mobile devices	Triada	Kouliaridis et al. (2020); Sallow et al. (2020) [37]
Wiper Malware	Erases user information in a way that makes it unrecoverable.	WhisperGate	Revay (2022) [51]

2.3. Multi-attack

Hu et al. (2022) [22], proposed a detection system for intrusions involving diverse combinations of attacks with multi-classification capacity based on the mosaic-coded convolution neural network. The one-dimensional CAN ID was transformed into a mosaic-like two-dimensional data grid for the CNN in order to efficiently extract the data characteristics and preserve the temporal relationships between the CAN IDs. They trained and tested the model using four different attack types in all feasible combinations. In order to simplify the model, the autoencoder was also utilised to decrease the data's dimensionality. The proposed method was successful in detecting all forms of attack combinations and had a high and stable multi-classification capacity.

Li et al. (2023) [75] use a CNN based on an auto-encoder to accomplish multi-attack detection, which ensures the detection accuracy of multi-attacks with the multiple classification function. They assessed the system using four different types of actual ICT attack data and four popular IDS techniques, and we showed that our framework outperformed all benchmarks in terms of accuracy, recall, precision, and F1-score. The finding showed a significant step toward developing an IDS that can identify multi-attacks in both intra-terminal and inter-terminal networks.

A framework for multiple strategy combinations is presented by Wang et al. (2020) [65]. Five different static feature types were used to categorise Android applications. They employ three filter-based feature selection techniques to obtain the top-k features that are most informative in order to increase classification accuracy and decrease overfitting. The applications represented by the feature subsets are then fed into five classification algorithms to create classifiers. Finally, the classification outcomes were estimated using either hard voting or soft voting. The experimental findings demonstrate that the method can achieve above 98 percent accuracy, precision, recall, and F-score. The technology offers the highest malware detection rate of 98.75 percent when compared to other approaches currently in use.

The method proposed by Da Costa and Moia (2023) [21] for detecting Android malware consists of a collection of specific-type detectors, each of which does a multi-stage analysis based on rules and ML techniques at various stages of the application cycle (before and after its installation). The strategy also differs from cutting-edge solutions in that it is non-invasive because it uses a method to get application functionalities without violating the licences and conditions of use of those programmes. The

findings showed that the concept offers a better identification method and is three times faster and, in some situations, ten times less CPU-intensive than other methods for classifying applications.

A multi-tiered feature selection technique is presented by Bhat and Dutta (2022) [16] , and it was used to find important features that will help malware detection methods be more accurate. To classify the chosen feature set, the proposed method uses five ML algorithms. The best static feature set (OSFS) and most significant features (MIFs) are determined by each ML method. Random Forest classification, which has a 96.28 percent accuracy rate, is the result of rigorous testing and research.

A Multi-level Anomaly Detector for Android Malware (MADAM) was proposed by Dini et al. (2012) [20]. MADAM uses ML techniques to discriminate between normal and dangerous actions while simultaneously monitoring Android at the kernel and user levels to find true malware infections. The first version of MADAM can recognise a variety of genuine malware that has been observed in the wild. Due to the limited number of false positives produced after the learning phase, MADAM has no impact on the usage of the device.

2.4. Deep Learning Methods

Android devices and network resources are harmed by malicious malware. Many deep learning algorithms are used for malware detection. Neural networks and other deep learning methods have recently been used in many different fields. The detection of Android malware has been effectively accomplished using deep learning models (Mahdavifar et al., 2020) [39]. The architectures utilise the unique characteristics of the domains they are used in to increase classification accuracy (Kaushik et al., 2022) [35].

2.4.1. Deep Neural Network (DNN)

Malware classification can be done using deep neural networks (DNNs), which are based on standard feed-forward neural network design (He et al., 2022). DNNs are usually feed-forward networks (FFNNs), in which data goes from the input layer to the output layer without going backward and the links between the layers are only ever in the forward direction (Singh et al., 2023) [57]. DNNs can be used to detect malware at different levels and are widely used in applications that require safety. DNNs have a huge number of parameters that are tuned by the training process in addition to an architecture that shares certain similarities with conventional software programmes (Miikkulainen et al., 2019) [41]. Any strategy for testing DNNs must take into account the special characteristics of DNNs, such as the semantic relationship between layers, the ReLU (Rectified Linear Unit) activation functions, and the syntactic connections between neurons in adjacent layers (Chen et al., 2020) [41].

DNNs are computational models made up of numerous small processing units (similar to neurons) stacked in interconnected layers and operating concurrently (Wichmann & Geirhos, 2023). Simple neural networks include two levels: an input layer and an output layer. Deep neural networks have multiple layers stacked on top of each other. Training is the process through which a DNN learns certain tasks and determines the strength of connections between its units (Thakkar & Lohiya, 2023). The trained DNN is then used to carry out the identical task on brand-new inputs. Studies have demonstrated that DNNs outperform all other models in predicting human perceptual similarity judgments and accounting for brain activity in primate sensory cortices (Miikkulainen et al., 2019; Chen et al., 2020) [41].

2.4.2. Recurrent Neural Network (RNN)

RNN functions better with input, which must be processed consecutively. The structural properties are present in the binary indicator vector utilised to represent Android applications (Kasongo et al., 2023) [34]. RNNs have the ability to process input classifications by using their internal state (memory). They can be used for tasks such as unsegmented script recognition or speech recognition (Wilberforce et al., 2023) [67]. RNNs are also susceptible to issues like the long-term dependency/vanishing gradient problem, in which information rapidly decays over time (Zhong et al., 2023) [74]. In fact, the neuron does not lose weight, whether it reaches a value of 0 or 1,000,000. However, since the weight is what stores the knowledge from the past in this situation, the preceding condition will not be very instructive (de Carvalho Junior et al., 2023) [24]. A time-based approach to FFNN is used by recurrent neural networks (RNN). The connections in this neural network span both passes and time, indicating that it is not stateless (Singh et al., 2023) [57].

2.4.3. Convolutional Neural Network (CNN)

CNN, a subclass of deep neural networks, is frequently used to examine visual data. They also have applications for speech recognition, video

interpretation, malware detection, and natural language processing. Sharma et al. (2019) [55]. introduce a 1-dimensional CNN-based malware detection solution that is extremely accurate and effective. When a binary file is input, the system determines if it is malicious or benign. The binaries undergo just a little pre-processing, and network training is used to uncover characteristics. The recursive learning process that occurs when a neural network algorithm identifies malware or develops a likelihood to render software "infected" The use of 1-dimensional convolutions distinguishes this method from previous CNN-based systems, and it provides important advantages for the detector.

Al-Rimy et al. (2020) [10], constructed a TFIDF-based standard malware detector in order to compare the proposed CNN detector with advanced methods. Experiments reveal enhanced accuracy of the proposed CNN detector while maintaining comparable training times. On a publicly accessible dataset of 11130 binaries, the system is also contrasted with the present embedding-based CNN detector. The approach performs better than embedding-based CNN in terms of accuracy and training time.

2.4.4. Multilayer Perceptron (MLP)

A feedforward artificial neural network class called a MLP was developed by Ben Abdel Ouahab et al. (2022) [1][2] for image-based malware classification, which consists of a number of fully connected layers. Every additional layer is made up of a group of nonlinear functions that represent the weighted total of all the outputs from the layer before it, all of which are connected. They experiment with various topologies by altering hidden layers, neurons, and activation functions in order to achieve a high degree of accuracy. The experiment achieved a precision of 97.6%. The multi-layer perceptron technique is an effective malware classifier with the chosen hyperparameters.

Singh and Singh (2020) [56] employed an MLP model with dynamic characteristics to binary classify benign and malicious files. The Cuckoo sandbox is used to execute both malicious and benign samples in the dynamic analysis environment. The thorough behavioural reports produced by Cuckoo Sandbox include a variety of runtime features such as API calls, registry modifications, and network activity. These features are looked at and signified as a feature set for training. After that, the training feature set is used to train a multi-layer perceptron model. The multi-layer perceptron model is trained using various activation functions, loss functions, and alpha parameter values in order to create the best malware classifier. The suggested malware classifier achieved a binary classification accuracy of 99.2 percent using the Adam loss function.

2.4.5. End-to-end (E2E)

E2E is a subset of DL that eliminates manual feature engineering by employing DL models to transform the raw inputs into the required outputs and predictions (Novikova et al., 2017) [46]. E2E DL has recently gained a lot of traction in practically all cutting-edge AI applications. This method has shown best-in-class results when applied to malware classification and detection (Almomani et al., 2023). The domain of virus detection in portable executables (PE) has demonstrated good potential for E2E DL architectures (Velasco et al., 2021). However, the DL classification has been applied less liberally and not in an E2E way in the case of Android malware.

An effective end-to-end ransomware detection system (E2E-RDS) is presented by Almomani et al. (2023) [9] that fully combines the available ransomware detection (RD) techniques. As with static-based RD, E2E-RDS relies on reverse engineering the ransomware code to parse and extract the key elements for prediction. Additionally, just like with vision-based RD, E2E-RDS can take a ransomware executable, transform it into an image, and then analyze it. The collected characteristics from the static-based RD technique are sent to eight different ML models to evaluate how well they can detect objects. The binary executable files of both benign and ransomware apps are transformed into 2D visual pictures using the vision-based RD technique. Then, in order to distinguish between ransomware apps and legitimate apps, these photos are sent to 19 distinct CNN models using the significant benefits of fine-tuning (FT) and transfer learning (TL) techniques. The primary advantage of the vision-based method is that it can accurately and efficiently detect and identify ransomware without the need for data augmentation or laborious feature extraction procedures. A recently compiled balanced dataset made up of 500 benign and 500 ransomware apps was used to conduct extensive simulations and performance assessments utilising several evaluation criteria for the proposed E2E-RDS. The results show that, in comparison to other evaluated ML models, the static-based RD technique employing the Ada Boost model showed high classification accuracy, reaching 97 percent. While the classification accuracy of the vision-based RD

method was high, reaching 99.5% for the FT ResNet50 CNN model,

2.5. *Bagging/ bootstrap aggregation*

Multiple iterations of the same deep learning model are possible. The several iterations with varying degrees of enhancement can be trained using various datasets (Bose et al., 2023) [19]. The bagging method combines various iterations of the same deep learning models (Zhang et al., 2022) [72][73]. Bootstrapping and aggregation are combined to create an ensemble model, hence the name "bagging." Several bootstrapped subsamples are taken from an initial sample of data. A training dataset is randomly sampled and replaced, allowing for multiple selections of the same data points. The most effective predictor is created by combining each subsample with the others using an algorithm (Zhang et al., 2022) [72][73]. A simple average or weighted average is used to aggregate several iterations of the same deep learning model (Malek et al., 2023) [40]. With the use of this technique, a new model may be developed that lacks the confirmation bias that might develop with a single model, leading to a model that is more precise and effective (Bose et al., 2023) [19].

3. MATERIALS AND METHODS

3.1 *Datasets and Proposed Framework*

Android applications balance the collection of benign and malicious datasets for malware detection. Data gathering is a sample of the different malware families found in the wild. Table 1 depicts sources where APKs are gathered and combined to provide a comprehensive dataset. Android programmes are stored in Android Application Package Kit (APK) archive files and run on the Android operating system.

Table 1: Android

S/no.	Dataset	Description
1	AndroZOO	42,910 malware and 44,347 benign
2	CASANDRA	Repository for benign samples (258 packages)
3	AppsApk	5,560 malwares and 9,476 benign
4	F-Droid	9,133 malwares
5	APKPure	
6	DroidFusion	
7	Rmvdroid	
8	CICInves And Mal2019	
9	AMD Project	
10	CICAAGM	
11	CIC And Mal 2017	
12	Drebin	
13	AndroGBFS	
14	Android Botnet	
15	Contagio Data	

The proposed framework of this study for malware detection is displayed in Figure 1. We proposed five types of data collections based on the deep learning methods [i.e., malware multi-attack detectors (MMAD)] used in this study. The MMAD uses combined detectors comprised of DNN, RNN, CNN, MLP, and E2E. All five types of datasets consist of both malware and benign data. The datasets comprise hidden features and some types of malware. The datasets for DNN consist of rootkits, bots, and ransomware; RNN consists of adware, ransomware, and keyloggers; CNN consists of spyware, trojans, and mobile malware; MLP consists of wiper malware and worms; and E2E consists of fileless malware and worms. These models are independently trained and then combined using bagging or bootstrapping aggregate. The malware analysis is performed using the combined model, and the mode is used for detection accuracy.

Journal of Theoretical and Applied Information Technology
15th January 2024. Vol.102. No 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

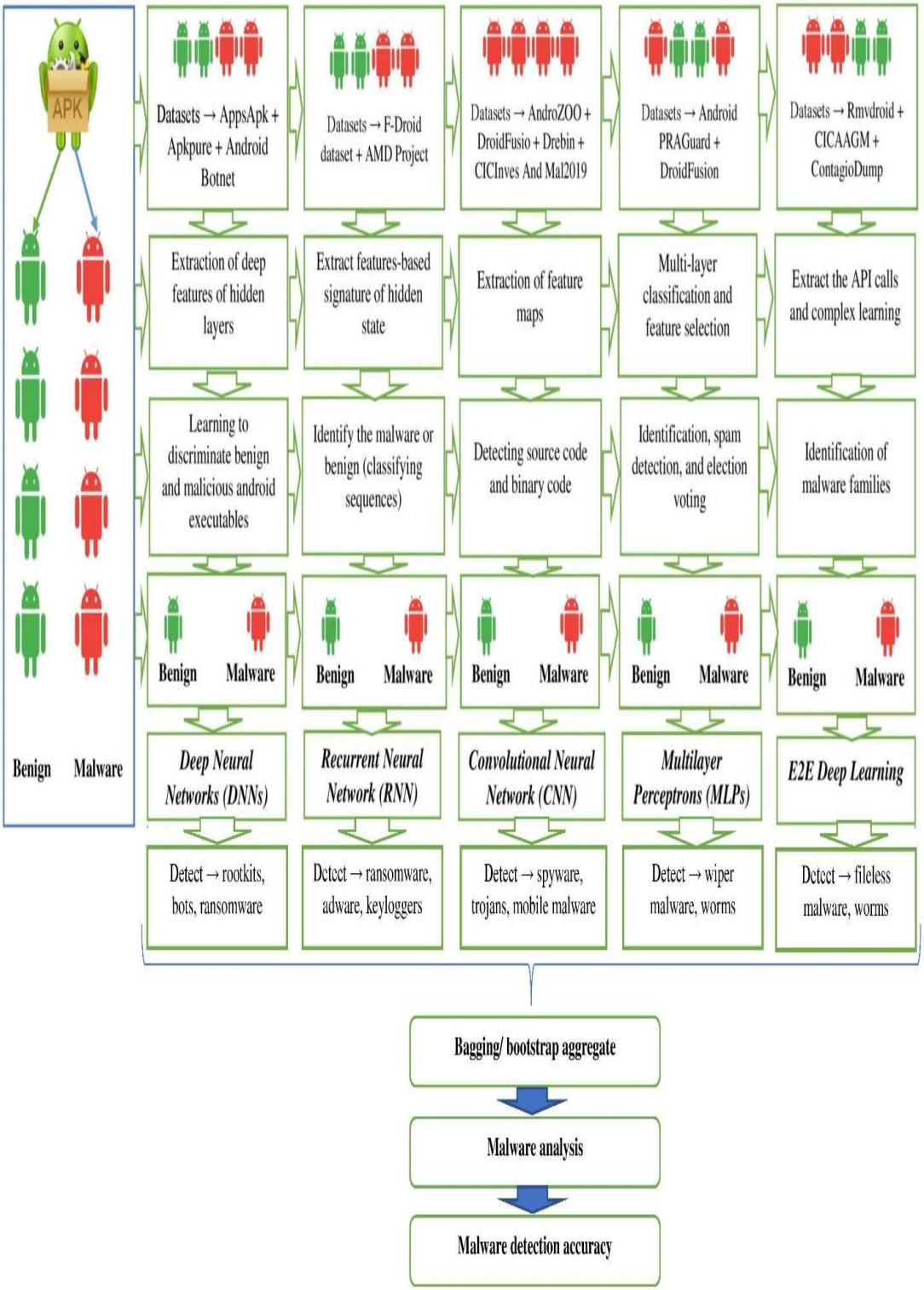


Figure 1: Proposed Method Of Malware Detection (MMAD)

3.2. Models

DNN model

This model estimates the behavior of any function. The output (y) of a component (i) in layer (l) is connected to the output (x) of the previous layer (k) with J outputs via a set of weights ($w_{i,k}$), a bias (b) and *f* as a non-linear activation function.

$$\text{DNN} = Y_1^l = f\ \sum_{j=1}^{J} w_{i,k}\, x_k + b_i \quad (1)$$

RNN model

Input: *x(m)* is considered as the input to the android at time *m*.

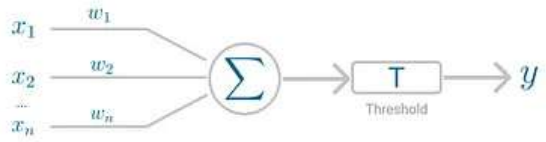
Output: *o(m)* represents the output of the android connection.

h(t) denotes a hidden state at time m and functions as "recall" of the android network. *h(m)* is computed based on the present input and the prior time's hidden state

$$\text{RNN} = \text{h(m)} = f\,[U\ \text{x(m)} + W\ \text{h(m}-1)] \quad (2)$$

The function *f* is considered as non-linear conversion including *tanh*, *ReLU,* etc. The RNN has input to hidden networks factorized by a weight matric U, hidden-to-hidden repeated networks factorized by a weight matric *W*, and hidden-to-output networks factorized by a weight matric V with all these matrices (*U*,*V*,*W)* are distributed throughout the time.

CNN model



The activation function is represented by the threshold T. The neuron outputs the value 1 if the weighted sum of the inputs is greater than 0; otherwise, the output value is 0. The gradient of the mean squared error is calculated across all input and output pairs in each iteration after the weighted sums have been passed through all layers. Given that inputs and initial weights are mixed in a weighted sum and are both subject to the activation function, MLP falls within the class of feedforward algorithms. However, the distinction is that each linear combination is carried over to the following layer.

The output of each layer's computation and inner representation of the data are fed to the layer below it. This passes through all hidden layers and ends at the output layer.

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume.

The input volume size *W*, the convolutional layer neurons' kernel field size *K*, the stride *S*, and the number of zero paddings *P* on the edge are all factors that affect the spatial size of the output volume. Therefore, the number of neurons that "fit" in a precise volume is:

$$= \frac{W-K+2P}{S} + 1 \quad (3)$$

The neurons cannot be covered to fit over the input volume symmetrically if this value is not an integer, indicating that the strides are inappropriate. Generally, when the stride is $S = 1$, fixing the zero padding to be $P = (K - 1)/2$ ensures that the input volume and output volume will be the same size spatially. Conversely, using every single neuron from the preceding layer is not always necessary.

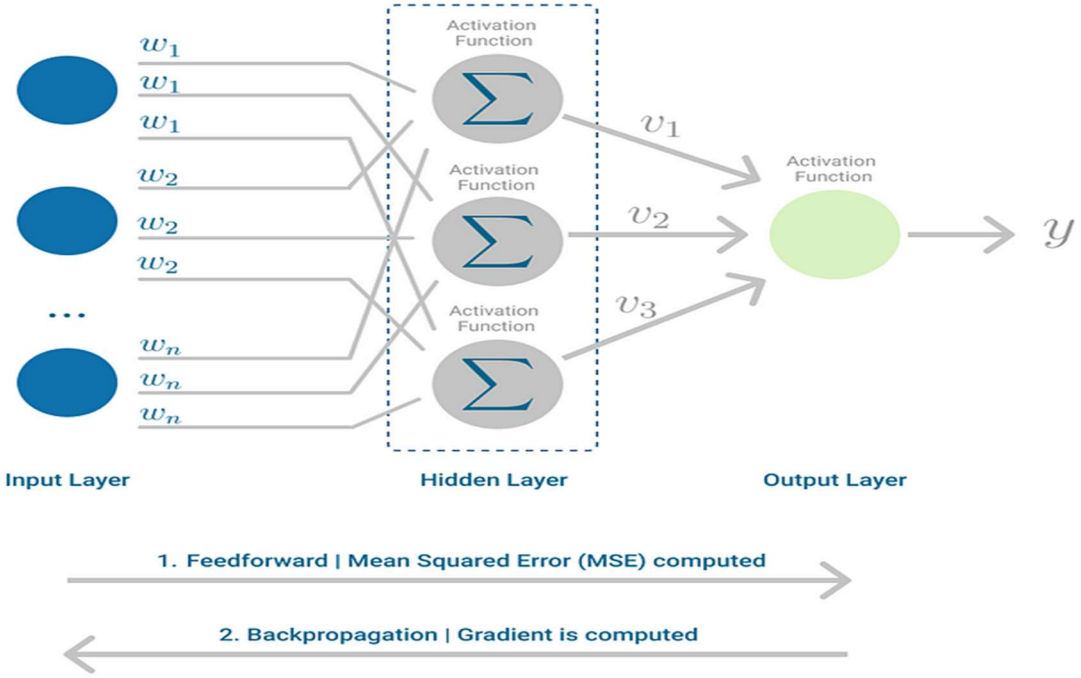
MLP model

An input, output, and one or more hidden layers—each with numerous neurons stacked together—make up a multilayer perceptron. The neurons in a MLP use an activation function that imposes a threshold, such as ReLU or sigmoid.

$$\text{F(x, w)} = x_1 w_1 + \cdots + x_n w_n \quad (4)$$

F(x, w) is output, $x_1 w_1$ is input, and $x_n w_n$ is the weights. The activation function is written as

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1}$$



E2E model

The target of the replenishment problem is to establish the finest order quantity, a: f(x) ∈ R at every specified point, through experimental features, x.

$$\sum_{i=1}^{N} L(f(x_i): a_i^*)$$

where N is the overall number of training data, L is the loss function described based on the difference between the model prediction $f(x_i)$ and the optimal order quantity a_i^*. Particularly, when examine neural network representations of function f.

3.3. Bagging/ bootstrap aggregation Method

Bagging, often referred to as "bootstrap aggregation," is frequently used to decrease variance in a dataset. Following the generation of several data samples, these models are independently trained for classification, and the average or regression of those predictions results in a more accurate estimate. During the combination operation, bagging greatly lowers an estimate's variance, improving the estimate's accuracy. Thus, compared to individual results, the learned results show more stability. This implies that a number of models are assembled, increasing the weights of the inaccurately categorised data in the independent model with each new model iteration. The algorithm may more easily focus on the factors that will help it perform better under this redistribution of weights. Parallel computations for each model are performed, and the results are then combined. For regression problems, the simple average is used to calculate the final ensemble aggregation, while for classification problems, the simple majority is used.

$$S_L(.) = \frac{1}{L}\sum_{l=1}^{L}\theta_l$$

simple average for regression problem

$$S_L(.) = \arg\max_k [card(l|\theta_l = k)]$$

Simple majority vote for classification problem

Classification model for bagging

```
#Create classification model for bagging
model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
```

Train models and print their accuracy

```
#Train different models and print their accuracy
results = model_selection.cross_val_score(model, X_fit, y_fit,cv=kfold)
for i in range(len(results)):
    print("Model: "+str(i)+" Accuracy is: "+str(results[i]))
```

```
Model: 0 Accuracy is: 1.0
Model: 1 Accuracy is: 1.0
Model: 2 Accuracy is: 1.0
Model: 3 Accuracy is: 0.9090909090909091
Model: 4 Accuracy is: 1.0
Model: 5 Accuracy is: 1.0
Model: 6 Accuracy is: 0.9
Model: 7 Accuracy is: 1.0
Model: 8 Accuracy is: 1.0
Model: 9 Accuracy is: 0.7
```

Print the mean accuracy

```
print("Mean Accuracy is: "+str(results.mean()))
```

```
Mean Accuracy is: 0.9509090909090908
```

3.4. Training of Models

The models are individually trained and then combined using bagging. Each model receives training using a unique dataset, as shown in Figure 1. The samples are generated by randomly choosing 2/3 of the replacement samples from the training sets. The hidden layer's number of neurons, "nh," and the activation function, "act," are the two parameters that must be set for the model. The following choices for activation functions are used:

1. Tanh
2. Sigmoid
3. ReLU
4. Tanh with Exponential
5. ReLU with Dropout

Although models are effective deep learning techniques, they can overfit. Dropout, a method for resolving this, involves randomly removing units from the network together with their incoming and outgoing connections. The components consequently do not co-adapt much.

3.5. Malware Analysis

This part presents the procedures needed to transform an executable into a form that may be used as input by a learning-based algorithm. Reverse engineering tools are used to convert the executable from deep learning code to an interpretable format, from which features are extracted to provide sequential features for learning.

3.5.1. Reverse engineering

Reverse engineering is used to decode the binary instructions and reveal the program's working logic. A variety of features can be determined by feeding the samples from the acquired dataset into the tools for reverse engineering the apks. The malware analysis is carried out to extract features from a sparse binary feature matrix, where the columns are the different features that were extracted. Dynamic analysis produces a list of API calls that are made in response to various system-related operations, including memory, process, file, and network activity.

3.5.2. Feature extraction

Many features are extracted directly from the .apk file for more accurate and effective malware analysis. The following features are collected from malware analysis for Android applications:

Application components: The four application components of an Android application are service, activity, content provider, and broadcast receiver. Without user input, service components operate in the background. Interfaces for activity components are provided. Using content providers facilitates data exchange between apps. Broadcast receivers handle system-wide announcements.

Filtered intents: The Android message transmission system uses intentions to communicate among its many parts. The filtered intents serve as an indication of each component's operation. Each application component registers itself to get intents using intent filters.

Data Flow: With reference to handling a large volume of datasets for data processing on an Android device, Data Flow Graph is utilised to perform offline malware detection. They are therefore less easily regarded as immediate on-

device detection models.

3.2. Experiment

The Android application (A) is composed of parts that connect to the internet at various levels. The overall behaviour of the Android system is made up of all of its components, and the malware can affect its defensive mechanism (Yadav et al., 2022). In the present study, malware incursions were efficiently detected using the combined model.

The combined model was used to perform malware analysis on the multi attacks. The Android with malware multi-attack represents the sensitive API calls, malicious permissions, privilege escalation attack, component intent communication, and sending sensitive information flow. The multi-attack approach's effect on the Android component is represented by ∃ComL(L ∈ n):

$$Android \coloneqq = \exists ComL \sum_{j=1}^{M} feature_j$$

$$\coloneqq = \sum_{j=1}^{M} feature_j \, [\hat{W}(x_1) \mid W(x_1)] \mid_1 (\text{feature})A! \; \text{Feature}_j$$

where

$\sum_{j=1}^{M} A_1$ featurej denotes activities of ComL

A_1 denotes malicious permissions (hidden malware) targeted at android component

x denotes sensitive information of the component

The malware features were extracted from tests A1, A2, A3, A4, and A5 (representing Android devices). The multi-attack model for the AFT was modelled by:

(1) DNN model is represented by the following algorithm:

$$\text{DNN} \coloneqq = [\hat{W}(x_1) \mid W(x_1)]$$

(2) RNN model is represented by the following algorithm:

$$\text{RNN} \coloneqq = \sum_{j=1}^{M} A_1.feature_j \mid [\hat{W}(x_1, x_2) \mid W(x_1, x_2)] \mid (\bar{\bar{U}}\psi) A_1$$

$$\coloneqq = A_1.feature_1 + A_1.feature_2 + A_1.feature_3 \mid [w(x_1) \mid \bar{w}(x_1)].A_1$$

(3) CNN model is represented by the following algorithm:

$$CNN \coloneqq = \sum_{j=1}^{M} A_1.feature_j \mid [\hat{W}(x_1, x_2) \mid W(x_1, x_2)] \mid (\bar{\bar{U}}\psi) A_1$$

$$\coloneqq = A_1.feature_1 + A_1.feature_2 + A_1.feature_3 + A_1.feature_4 \mid [w(x_1) \mid \bar{w}(x_1)].A_1$$

(4) MLP model is represented by the following algorithm:

$$\text{MLP} \coloneqq = \sum_{j=1}^{M} A_1.feature_j \mid [\hat{W}(x_1, x_2) \mid W(x_1, x_2)] \mid (\bar{\bar{U}}\psi) A_1$$

$$\coloneqq = A_1.feature_1 + A_1.feature_2 + A_1.feature_3 + A_1.feature_4 + A_1.feature_5 \mid [w(x_1) \mid \bar{w}(x_1)].A_1$$

(5) E2E model is represented by the following algorithm:

$$\text{E2E} \coloneqq = \sum_{j=1}^{M} A_1.feature_j \mid [\hat{W}(x_1, x_2) \mid W(x_1, x_2)] \mid (\bar{\bar{U}}\psi) A_1$$

$$\coloneqq = A_1.feature_1 + A_1.feature_2 + A_1.feature_3 + A_1.feature_4 + A_1.feature_5 + A_1.feature_6 \mid [w(x_1) \mid \bar{w}(x_1)].A_1$$

The model measures the android behavior of malware multi attack based on the following:

(1) The behavior of DNN model is shown as:

$$\text{DNN} \coloneqq = [w(data_1) \mid \bar{w}(data_1)].$$

(2) The behavior of RNN model is shown as:

$$\text{RNNmodel} \coloneqq = A_1.\text{AttackerIntent1} + A_1.\text{RNN} + A_1.\text{filterRNN} \; [w(data_1) \mid \bar{w}(data_1)].A_1.$$

(3) The behavior of CNN model is shown as:

$$\text{CNNmodel} \coloneqq = A_1.\text{AttackerIntent2} + A_1.\text{CNN} + A_1.\text{filterCNN} + A_1.\text{HiddenCNN} \; [w(data_1) \mid \bar{w}(data_1)].A_1.$$

(4) The behavior of MLP model is shown as:

$$\text{MLPmodel} \coloneqq = A_1.\text{AttackerIntent2} + A_1.\text{MLP} + A_1.\text{filterMLP} + A_1.\text{HiddenMLP1} + A_1.\text{HiddenMLP2} \; [w(data_1) \mid \bar{w}(data_1)].A_1.$$

(5) The behavior of E2E model is shown as:

$$\text{E2Emodel} \coloneqq = A_1.\text{AttackerIntent2} + A_1.\text{E2E} + A_1.\text{filterE2E} + A_1.\text{HiddenMLP1} + A_2.\text{HiddenMLP2} +$$

A_3.HiddenMLP3 [w ($data_1$) | w̄ ($data_1$)].A_1.

4. RESULTS

4.1. Bagging/ bootstrap aggregation

The results obtained from bagging and bootstrap aggregation are presented in Table 2. The bagging had the maximum accuracy for test data, which is consistent with the outcomes found in the case of the bootstrap. Bagging and bootstrapping had the lowest standard deviation for both training and test data and an average accuracy rate of more than 70%. No deep learning approach had an average prediction accuracy less than 70% for the three middle layers, while the two middle layers had an average prediction accuracy greater than 70%. The ratio difference between training and test data (85% to 15%) in the three and middle layers had no significant effect on the results, thus causing no detectible variation in the prediction accuracy.

Table 2: Prediction accuracy (the ratio of training and test data is 85% to 15%) of the combined deep learning methods.

			Data	Ratio of Training and Test Data (%)	Accuracy ratio of training data		Accuracy ratio of test data		Area under the curve
	Method				**Average (%)**	**standard deviation**	**Average (%)**	**standard deviation**	
	Bagging		Normalized	85:15	73.58	1.45	70.98	1.55	0.877
	Bootstrap		Normalized	85:15	70.66	1.23	69.65	1.43	0.834
			Data	**Ratio of Training and Test Data (%)**	**Accuracy ratio of training data**		**Accuracy ratio of test data**		**Area under the curve**
Model	**Method Activation function**	**Middle layer**			**Average (%)**	**standard deviation**	**Average (%)**	**standard deviation**	
DNN	Tanh	3	Normalized	85:15	69.22	1.59	70.75	1.45	0.861
RNN	Tanh or sigmoid	3	Normalized	85:15	69.47	1.38	70.55	1.31	0.863
CNN	ReLU	3	Normalized	85:15	61.99	1.50	60.89	0.99	0.958
MLPs	ReLU	3	Normalized	85:15	66.78	1.49	65.80	1.70	0.911
E2E	ReLU	3	Normalized	85:15	67.89	2.02	70.00	1.60	0.850
DNN	Tanh with Exponential	2	Normalized	85:15	70.77	1.67	69.45	1.50	0.858
RNN	Tanh with Exponential	2	Normalized	85:15	70.11	1.41	70.35	0.95	0.871
CNN	ReLU with Dropout	2	Normalized	85:15	70.23	2.08	70.66	1.40	0.801
MLPs	ReLU with Dropout	2	Normalized	85:15	70.91	1.08	69.85	1.64	0.809
E2E	ReLU with Dropout	2	Normalized	85:15	70.00	1.42	70.25	0.98	0.872

4.2. The Model

A MMAD detector can be installed on an Android device once it has been correctly constructed. MMAD can work with one or more detectors, each of which is designed to find specific types of malware in applications (see Figure 1). Figure 2 depicts the complete MMAD workflow with numerous detectors based on the specificity of the deep learning methods. The MMAD detectors installed on the device receive the application as input. We underline that all of the detectors function in parallel to assess a particular malware attack.

Given that the detectors are run in sequence, MMAD is first provided as input for Detector 1. Detector 1 gathers an initial collection of features relevant to the specific analysis of the target malware using native Android components. The detector then conducts an analysis of the APK file to detect relevant malware. In a successful situation, the application is sent to Detector 1's deep analysis component to continue the analysis. The APK is sent to the next detector (Detector 2), which continues the process using its unique set of rules (algorithms) and components, comparing them to a potential new set of features extracted from the APK. If no malicious malware is detected, the

analysis is deemed safe, and the APK can now be loaded on the device without further delay.

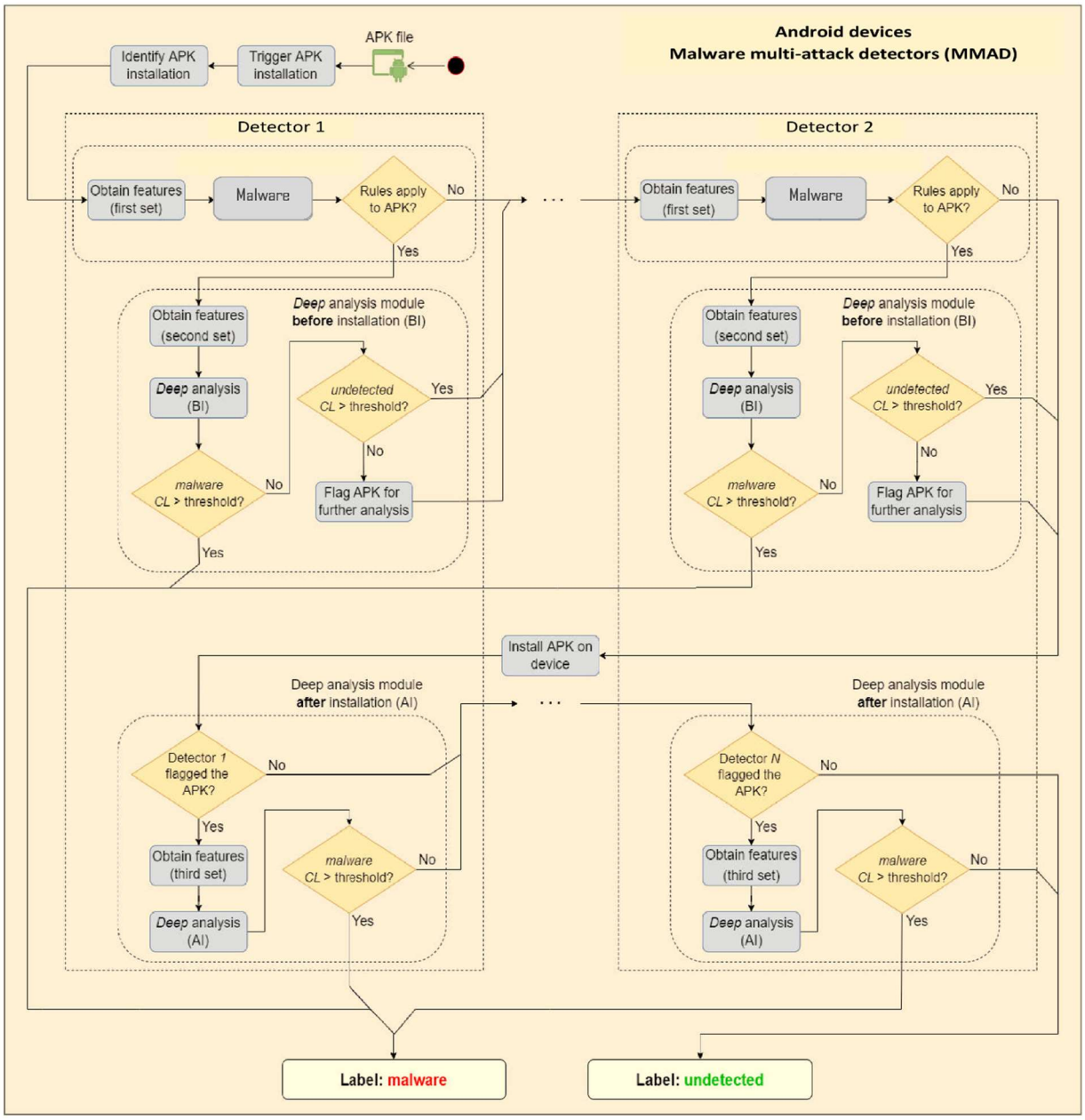


Figure 2: The Malware Multi-Attack Detectors (MMAD)

4.3. Multi-attack layer Detections

Table 3 displays the MMAD one hidden layer detection for diverse size ranges from 56 neurons to 254 neurons. The analysis using layer sizes more than 254 neurons was not tested because there were only 389 different malware attacks. As the embedding size was maintained at 56, any hidden layer with less than 56 neurons was not assessed. In order to evaluate any configuration with more than one hidden layer in subsequent rounds, the hidden layer size that indicated equivalent performance with the fewest neurons was fixed as the upper cap. This layer appeared to detect lesser numbers of malware despite the presence of many neurons.

Table 3: MMAD one hidden layer detection

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Conf. ID	Input	Output	Hidden Layer Size	Embedding Dimension	Learning rate	Detection level	Number of Malware
1	33.67	70.23	[139]	254	85.016	60.79	389
2	32.52	71.19	[139]	200	85.017	60.50	290
3	33.15	70.35	[139]	190	85.015	60.79	248
4	33.89	70.17	[139]	250	85.018	60.79	376
5	35.60	70.58	[125]	190	85.017	60.79	248
6	33.91	71.99	[125]	184	85.018	60.79	66
7	34.08	70.45	[125]	120	85.018	60.79	113
8	33.73	70.65	[72]	70	85.016	60.79	53
9	35.11	72.12	[72]	85	85.018	60.79	133
10	34.66	70.20	[65]	65	85.018	60.50	178
11	33.94	71.33	[65]	80	85.019	60.78	101
12	33.94	71.33	[56]	90	85.018	60.78	112
13	33.94	71.33	[56]	100	85.019	60.78	188

Table 4 presents the results of MMAD two-hidden-layer detections. In the second phase, MMAD was tested for two hidden layers. Data complexity and linearity contribute to the DL method's ideal performance on the training datasets. Layer sizes beyond 139 neurons were not considered since the requirement for a single hidden layer within the size of a hidden layer is more indicative of the linearity of the two-hidden layer. The first hidden layer's size varies between 139 and 56, while the second hidden layer's size could vary between 139 and 72. The MMAD gradually reduced the data dimension cardinality.

Table 4: MMAD Two-Hidden-Layer Detections

Conf. ID	Input	Output	Hidden Layer Size	Embedding Dimension	Learning rate	Detection level	Number of Malware
14	33.89	70.17	[139, 139]	74	85.011	75.21	1173
15	35.60	70.58	[139, 72]	74	85.010	75.65	956
16	33.91	71.99	[125, 65]	74	85.019	75.68	1160
17	34.08	70.45	[125, 56]	74	85.021	75.90	1098
18	33.73	70.65	[72, 56]	74	85.025	75.31	981
19	35.11	72.12	[65, 56]	74	85.021	75.45	879

We examined numerous configurations with three hidden layers of various sizes in the third iteration of the MMAD detection. Table 5 presents the results of MMAD three-hidden-layer detection. The results showed consistency and higher performance than one hidden layer and two hidden layers. It was robust in detecting higher-level multi-attack malware. The limitations for the first hidden layer's maximum number of layers and any future hidden

layers' equivalent constraints remained the same as in the second iteration. The MMAD with more than three hidden layers was not analysed because none of these configurations outperformed some other configurations with only two hidden layers that were evaluated in earlier iterations.

Table 5: MMAD three-hidden-layer detection

Conf. ID	Input	Output	Hidden Layer Size	Embedding Dimension	Learning rate	Detection level	Number of Malware
20	33.89	70.17	[139, 72, 56]	74	85.042	90.82	1540
21	35.60	70.58	[139, 125, 72]	74	85.042	90.77	1413
22	33.91	70.99	[139, 125, 65]	74	85.041	90.46	1608
23	34.08	70.45	[139, 65, 56]	74	85.045	90.12	1371
24	33.73	70.65	[139, 72, 65]	74	85.043	90.19	1447

4.4. MMAD Detection Accuracies and performances

The MMAD was subjected to experiments with various numbers of hidden layers and various sizes of each of these hidden layers to determine their accuracy in detecting malware. Table 6 presents the results of the MMAD accuracies based on their optimal hidden layers. According to the results, all the hidden layers produced higher accuracies with optimal F1, which indicated better performance of the MMAD in detecting malware. There were no significant variations between the DL methods in terms of accuracy. Based on 1 to 3 hidden layers with 56 to 139 neurons per hidden layer, MMAD was able to detect the multi-attacks with accuracy. We were able to avoid evaluating many configurations that were less likely to yield better outcomes than similar configurations that also had lower hidden outlays.

Table 6: The MMAD accuracies based on their optimal hidden layers

Conf. ID	Hidden Layer Size	Accuracy (Th=0.5)	Optimal F1	DNN Accuracy @ F1	RNN Accuracy @ F1	CNN Accuracy @ F1	MLP Accuracy @ F1	E2E Accuracy @ F1
1	[139]	0.844	0.231	0.641	0.673	0.516	0.635	0.581
2	[125]	0.714	0.233	0.639	0.672	0.515	0.633	0.585
3	[72]	0.852	0.233	0.674	0.675	0.517	0.637	0.572
4	[65]	0.747	0.200	0.674	0.663	0.552	0.574	0.552
5	[56]	0.874	0.237	0.675	0.678	0.564	0.634	0.459
6	[65, 65]	0.614	0.252	0.645	0.677	0.538	0.644	0.558
7	[125, 125]	0.844	0.241	0.646	0.681	0.551	0.646	0.551
8	[139, 139]	0.843	0.251	0.644	0.675	0.585	0.644	0.553
9	[125, 125, 125]	0.848	0.212	0.652	0.588	0.533	0.651	0.544
10	[65, 65, 65]	0.838	0.214	0.652	0.588	0.411	0.646	0.562
11	[125, 125, 125, 125]	0.837	0.289	0.614	0.601	0.489	0.586	0.499
12	[72, 72, 72, 72]	0.837	0.211	0.658	0.592	0.517	0.657	0.529

13	[125, 125, 65, 65]	0.836	0.216	0.651	0.688	0.436	0.649	0.515
14	[65, 65, 65, 65]	0.836	0.226	0.675	0.588	0.437	0.645	0.561
15	[56, 56, 56, 56]	0.846	0.239	0.614	0.635	0.503	0.686	0.584

We carried out additional performance analysis to further validate MMAD detection efficiency. These statistics and the accuracy assessment of the validation data serve as the measures for evaluating MMAD performance. Table 7 displays the results of the detection efficiency of MMAD. The MMAD for static and active based on DL produced 96.54% accuracy (average). The 3-hidden layer outperformed the 2-hidden layer in all parameters. Additionally, all detectors used performed efficiently with excellent F-score values. The MMAD classification system efficiently distinguished between benign and malware, as well as the types of malware, with high accuracies and precisions across all neurons. The class probabilities and a threshold of 0.5 were used to differentiate between the classification and the validation accuracy. With average results of 96.54% accuracy, 95.38% precision, 92.65% recall, and a 94.66 F-score, the MMAD proved to be efficient and robust.

Table 7: Detection efficiency of MMAD

MMAD Classification System	Extracted Features	Detection			No. of neurons	Acc.	Prec.	Recall	F-score
		Benign	Malware	Types of Malware					
DNN [139, 56]	Static only	614	927	Rootkits, bots, ransomware	[72,72]	95.79	95.39	89.57	92.31
RNN [139, 56]	Active only	614	927	Ransomware, adware, keyloggers	[72,64]	94.6	95.59	88.25	91.99
CNN [139, 56]	Static & Active	614	927	Spyware, trojans, mobile malware	[72,64]	94.76	95.78	88.76	92.7
MLPs [139, 56]	Static (opcode)	660	1,100	Wiper malware, worms	[72,56]	94.37	95.56	87.9	92.9
E2E [139, 56]	Static only	652	1,017	Fileless malware, worms	[72,56]	95.21	96.02	89.08	92.25
DNN [139, 125, 72]	Active (sys. Calls)	4,138	7,840	Rootkits, bots, ransomware	[139,139,139]	98.65	95.96	93.11	98.68
RNN [139, 125, 72]	Active only	4,138	7,840	Ransomware, adware, keyloggers	[139,139,139]	97.99	94.08	97.17	95.33
CNN [139, 125, 72]	Static & Active	4,138	7,840	Spyware, trojans, mobile malware	[139,125,139]	97.42	95.36	97.21	96.11
MLPs [139, 125, 72]	Active only	4,138	7,840	Wiper malware, worms	[139,139,72]	98.01	94.9	97.25	96.56
E2E [139, 125, 72]	Static & Active	4,138	7,840	Fileless malware, worms	[139,139,64]	98.55	95.11	98.15	97.81
Average						96.54	95.38	92.65	94.66

The impact of increasing the training epoch count was then investigated. Table 8 displays the result of increasing the number of epochs in MMAD. The default setting of 100 epochs was increased to 1000 epochs. This is because the 1,000 epochs imply that the entire dataset flows through the MMAD model, which runs through it 1.000 times. The training accuracy showed higher accuracy and performance with more than 1000 epochs.

Table 8: Impact Of Increasing The Number Of Epochs In MMAD

Conf. ID	MMAD Hidden Layer Size	MMAD Epochs	MMAD AUC	Accuracy (Th=0.5)	FPR (Th=0.5)
16	[72, 72, 72, 72]	100	0.787	0.766	0.221
17	[72, 72, 72, 72]	1000	0.949	0.764	0.219

Note: AUC = Area Under Curve; FPR = False positive rate; Th = threshold (0.5)

The optimizer algorithm is crucial for analysing training's efficiency and efficacy (as measured by performance indicators). This is done to ensure the MMAD model's validation. We tested a few DL optimizer techniques (Wang et al., 2019) [64]. Adadelta was the standard optimizer algorithm used. In addition, we tested the adam and rmsprop optimizers (Solanke & Patnaik, 2020; Wu et al., 2022) while maintaining the other parameters. The result showed higher accuracies and a lower FPR.

Table 9: Impact of alteration in optimizer of MMAD

Conf. ID	MMAD Hidden Layer Size	MMAD Epochs	MMAD Optimizer	MMAD AUC	Acc (Th = 0.5)	FPR (Th = 0.5)
18	[72, 72, 72, 72]	1000	adadelta	0.936	0.844	0.125
19	[72, 72, 72, 72]	1000	adam	0.931	0.846	0.128
20	[72, 72, 72, 72]	1000	rmsprop	0.927	0.830	0.126

Note: AUC = Area Under Curve; FPR = False positive rate; Th = threshold (0.5)

Batch size is another hyperparameter that is directly related to both efficacy and efficiency. The standard batch size obtained was greater than 1,500 with consistent accuracies. We experimented with various batch sizes in the following iteration, as indicated in Table 10. Therefore, about six steps of training passes were carried out for each epoch for about 81902 training records. There was accuracy in the learning rate and a low FPR, which indicates the robustness of the MMAD model.

Table 10: Impact of alteration in MMAD batch size

Conf. ID	MMAD Hidden Layer Size	MMAD Batch Size	MMAD Optimizer	MMAD AUC	Learning rate	Acc (Th = 0.5)	FPR (Th = 0.5)
21	[72, 72, 72, 72]	2097	rmsprop	0.911	0.0011	0.775	0.123
22	[72, 72, 72, 72]	2310	rmsprop	0.909	0.0017	0.770	0.128

23	[72, 72, 72, 72]	1585	rmsprop	0.825	0.0014	0.765	0.131

Note: FPR = False positive rate; Th = threshold (0.5)

5. DISCUSSION

Our findings demonstrate the importance of combined DL methods to improve malware multi-attack detection (MMAD) on Android-based devices. To the best of our knowledge, MMAD is the first model to combine highly sensitive DL methods for Android with the capability of detecting various types of malware. Interestingly, it is vital to explain how the user can differentiate between a true intrusion and a false positive, which may be a concern. Following the MMAD learning phase, sporadic false positives decrease in frequency, and occasionally, detection may be linked to them. In reality, every type of malware that has been evaluated demonstrates aggressive tendencies that result in repeated and multiple detections within the third hidden layer of neurons. This framework might be extended to the automatic management of rare false positives or to guide the user via a smart learning phase so MMAD can rapidly learn during the detection process. When an attack is initiated on the Android device as a result of a new installation, the MMAD can be used to initiate a new learning period for detection. Thus, this is the way forward for modern cybersecurity toward Android devices.

The level of detection techniques and detection outcomes is superior to those of earlier android anomaly-based detection systems (Dini et al., 2012 [25]; Da Costa and Moia, 2023 [21] ; Islam et al., 2023) [32]. It is crucial to note that, in contrast to past methods (Wang et al., 2020) [65] , the MMAD model performs multi-attack detection at multi-levels with specific target malware identifications. This technique may be more successful at detecting rapid Android behavioural changes. For instance, it may be possible to mislead malicious application-specific controls through MMAD hidden-layer detections based on DL methods when used independently, but they function robustly together to detect multiple malwares.

Our results with 96.54% accuracy, 95.38% precision, 92.65% recall, and a 94.66 F-score showed that the MMAD approach is effective, efficient, and simple to use to detect eleven types of malware. These malwares are used in multi-attacks or combinatorially with high aggressive intent to hijack, harm, or steal information from Android devices. The validation tests demonstrated that the MMAD approach is capable of effectively identifying the security concerns associated with multi-attacks, such as the transfer of sensitive data and components. The validation tests recognize the components, malicious permissions, sensitive API calls, and the intent filter using the detection method proposed (Solanke & Patnaik, 2020 [58]; Wu et al., 2022) [68]. However, these methods do not allow for the detection of sensitive information transfers. In order to provide adequate model coverage to activate malicious behaviours during the active analysis of Android apps, test input creation is required. MMAD can use a variety of test input generation techniques, including static, active (malicious with dynamism), etc. The static approach is the most well-liked approach to input creation and has been heavily utilised by researchers (Alzaylaee et al., 2020; Almomani et al., 2023 [9]; Singh et al., 2023) [57].

6. 6. CONCLUSION

This work provides MMAD as a framework that combines DL, including DNN, RNN, CNN, MLP, and E2E, for malware multi-attack detection on Android devices. First, MMAD was combined using bagging/bootstrap. Second, diverse datasets from diverse sources containing both malware and benign sources were used to model MMAD. Third, the model was trained, followed by malware detection and analysis. The experimental results showed that MMAD had 96.54% accuracy and 95.38% precision in detecting different types of malware. This supports precisely locating malicious attacks with a minimum FPR. We showed that MMAD performed efficiently in terms of accuracy at three-hidden-layer detection. The results clearly showed that MMAD achieved great accuracy, outperforming the DL methods individually. To the best of our knowledge, MMAD is the first model to combine highly sensitive DL methods for Android with the ability to detect different types of malware. Moreover, MMAD significantly reduced the number of FPRs by three times during simultaneous malware detection.

A future study should include a thorough analysis of the effects of concept drift on the classification of Android malware as well as modifications to the threshold in MMAD. The possibility of self-

adaptation for MMAD could be investigated as a way to boost the effectiveness of DL approaches for Android malware detection. Ant colony, or greedy algorithm, can be used as a replacement for batch so that it can automatically adapt to malware multi-attack development and population drift. Using additional DL, such as radial basis function networks (RBFNs), self-organizing maps (SOMs), and long short-term memory networks (LSTMs), to carry out more thorough detection would be an easy way to extend the MMAD model in this direction.

REFERENCES

[1] Abdel Ouahab IB, Bouhorma M, El Aachak L, Boudhir AA. Towards a new cyberdefense generation: proposition of an intelligent cybersecurity framework for malware attacks. Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science). 2022 Oct 1;15(8):1026-42.

[2] Abdel Ouahab IB, Bouhorma M, El Aachak L, Boudhir AA. Towards a new cyberdefense generation: proposition of an intelligent cybersecurity framework for malware attacks. Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science). 2022 Oct 1;15(8):1026-42.

[3] Akhtar MS, Feng T. Malware Analysis and Detection Using Machine Learning Algorithms. Symmetry. 2022 Nov 3;14(11):2304.

[4] Alahmadi BA, Mariconti E, Spolaor R, Stringhini G, Martinovic I. BOTection: Bot detection by building Markov Chain models of bots network behavior. InProceedings of the 15th ACM Asia Conference on Computer and Communications Security 2020 Oct 5 (pp. 652-664).

[5] Alamro H, Mtouaa W, Aljameel S, Salama AS, Hamza MA, Othman AY. Automated Android Malware Detection Using Optimal Ensemble Learning Approach for Cybersecurity. IEEE Access. 2023 Jul 11.

[6] Alani MM, Awad AI. AdStop: Efficient flow-based mobile adware detection using machine learning. Computers & Security. 2022 Jun 1;117:102718.

[7] Albakri A, Alhayan F, Alturki N, Ahamed S, Shamsudheen S. Metaheuristics with Deep Learning Model for Cybersecurity and Android Malware Detection and Classification. Applied Sciences. 2023 Feb 8;13(4):2172.

[8] Aldhyani TH, Alkahtani H. Attacks to automatous vehicles: A deep learning algorithm for cybersecurity. Sensors. 2022 Jan 4;22(1):360.

[9] Almomani I, Alkhayer A, El-Shafai W. E2E-RDS: Efficient End-to-End Ransomware Detection System Based on Static-Based ML and Vision-Based DL Approaches. Sensors. 2023 May 4;23(9):4467.

[10] Al-Rimy BA, Maarof MA, Alazab M, Alsolami F, Shaid SZ, Ghaleb FA, Al-Hadhrami T, Ali AM. A pseudo feedback-based annotated TF-IDF technique for dynamic crypto-ransomware pre-encryption boundary delineation and features extraction. IEEE Access. 2020 Jul 29;8:140586-98.

[11] Alzaylaee MK, Yerima SY, Sezer S. DL-Droid: Deep learning based android malware detection using real devices. Computers & Security. 2020 Feb 1;89:101663.

[12] Aslan Ö, Aktuğ SS, Ozkan-Okay M, Yilmaz AA, Akin E. A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. Electronics. 2023 Mar 11;12(6):1333..

[13] Ben Abdel Ouahab I, Elaachak L, Bouhorma M. Image-based malware classification using multi-layer perceptron. InNetworking, Intelligent Systems and Security: Proceedings of NISS 2021 2022 (pp. 453-464). Springer Singapore.

[14] Bhandari G, Lyth A, Shalaginov A, Grønli TM. Distributed Deep Neural-Network-Based Middleware for Cyber-Attacks Detection in Smart IoT Ecosystem: A Novel Framework and Performance Evaluation Approach. Electronics. 2023 Jan 6;12(2):298.

[15] Bhardwaj A, Goundar S. Keyloggers: silent cyber security weapons. Network Security. 2020 Feb;2020(2):14-9.

[16] Bhat P, Dutta K. A multi-tiered feature selection model for android malware detection based on Feature discrimination and Information Gain. Journal of King Saud University-Computer and Information Sciences. 2022 Nov 1;34(10):9464-77.

[17] Bhavitha S, Rao MC, Nancharaiah P, Suhasini S. Continuous Digital System Analysis in different System Softwares Using Keyloggers to Validate the need of Security. In2023 International Conference on Computer Communication and Informatics (ICCCI) 2023 Jan 23 (pp. 1-7). IEEE.

[18] Bhavitha S, Rao MC, Nancharaiah P, Suhasini S. Continuous Digital System Analysis in different System Softwares Using Keyloggers to Validate the need of Security. In2023 International Conference on Computer

Communication and Informatics (ICCCI) 2023 Jan 23 (pp. 1-7). IEEE.
[19] Bose SS, Natarajan R, HL G, Flammini F, Praveen Sundar PV. Iterative Reflect Perceptual Sammon and Machine Learning-Based Bagging Classification for Efficient Tumor Detection. Sustainability. 2023 Mar 4;15(5):4602.
[20] Chen Y, Xie Y, Song L, Chen F, Tang T. A survey of accelerator architectures for deep neural networks. Engineering. 2020 Mar 1;6(3):264-74.
[21] Da Costa L, Moia V. A Lightweight and Multi-stage Approach for Android Malware Detection using Non-invasive Machine Learning Techniques. IEEE Access. 2023 Jul 18.
[22] Dameff C, Tully J, Chan TC, Castillo EM, Savage S, Maysent P, Hemmen TM, Clay BJ, Longhurst CA. Ransomware attack associated with disruptions at adjacent emergency departments in the US. JAMA network open. 2023 May 1;6(5):e2312270-.
[23] De Arroyabe IF, Arranz CF, Arroyabe MF, de Arroyabe JC. Cybersecurity capabilities and cyber-attacks as drivers of investment in cybersecurity systems: A UK survey for 2018 and 2019. Computers & Security. 2023 Jan 1;124:102954.
[24] de Carvalho Junior A, Angelico BA, Justo JF, de Oliveira AM, da Silva Filho JI. Model reference control by recurrent neural network built with paraconsistent neurons for trajectory tracking of a rotary inverted pendulum. Applied Soft Computing. 2023 Jan 1;133:109927.
[25] Dini G, Martinelli F, Saracino A, Sgandurra D. MADAM: a multi-level anomaly detector for android malware. In International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security 2012 Oct 17 (pp. 240-253). Berlin, Heidelberg: Springer Berlin Heidelberg.
[26] Du J, Raza SH, Ahmad M, Alam I, Dar SH, Habib MA. Digital Forensics as Advanced Ransomware Pre-Attack Detection Algorithm for Endpoint Data Protection. Security and Communication Networks. 2022 Jul 6;2022:1-6.
[27] Florackis C, Louca C, Michaely R, Weber M. Cybersecurity risk. The Review of Financial Studies. 2023 Jan 1;36(1):351-407.
[28] Gibert D, Mateu C, Planes J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. Journal of Network and Computer Applications. 2020 Mar 1;153:102526.
[29] He Z, Rezaei A, Homayoun H, Sayadi H. Deep neural network and transfer learning for accurate hardware-based zero-day malware detection. InProceedings of the Great Lakes Symposium on VLSI 2022 2022 Jun 6 (pp. 27-32).
[30] Hu R, Wu Z, Xu Y, Lai T, Xia C. A multi-attack intrusion detection model based on Mosaic coded convolutional neural network and centralized encoding. Plos one. 2022 May 5;17(5):e0267910.
[31] Hu R, Wu Z, Xu Y, Lai T. Multi-attack and multi-classification intrusion detection for vehicle-mounted networks based on mosaic-coded convolutional neural network. Scientific Reports. 2022 Apr 15;12(1):6295.
[32] Islam R, Sayed MI, Saha S, Hossain MJ, Masud MA. Android malware classification using optimum feature selection and ensemble machine learning. Internet of Things and Cyber-Physical Systems. 2023 Jan 1;3:100-11.
[33] Kanaker H, Karim NA, Awwad SA, Ismail NH, Zraqou J. Trojan Horse Infection Detection in Cloud Based Environment Using Machine Learning. International Journal of Interactive Mobile Technologies. 2022 Dec 15;16(24).
[34] Kasongo SM. A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. Computer Communications. 2023 Feb 1;199:113-25.
[35] Kaushik D, Garg M, Gupta A, Pramanik S. Application of machine learning and deep learning in cybersecurity: An innovative approach. InAn Interdisciplinary Approach to Modern Network Security 2022 May 2 (pp. 89-109). CRC Press.
[36] Kaushik P. Unleashing the Power of Multi-Agent Deep Learning: Cyber-Attack Detection in IoT. International Journal for Global Academic & Scientific Research. 2023 Jun 30;2(2):23-45.
[37] Kouliaridis V, Barmpatsalou K, Kambourakis G, Chen S. A survey on mobile malware detection techniques. IEICE Transactions on Information and Systems. 2020 Feb 1;103(2):204-11.
[38] Li B, Hu W, Qu X, Li Y. A Novel Multi-Attack IDS Framework for Intelligent Connected Terminals Based on Over-the-Air Signature Updates. Electronics. 2023 May 17;12(10):2267.
[39] Mahdavifar S, Kadir AF, Fatemi R, Alhadidi D, Ghorbani AA. Dynamic android malware category classification using semi-supervised deep learning. In2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence

and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech) 2020 Aug 17 (pp. 515-522). IEEE.

[40] Malek NH, Yaacob WF, Wah YB, Md Nasir SA, Shaadan N, Indratno SW. Comparison of ensemble hybrid sampling with bagging and boosting machine learning approach for imbalanced data. Indones. J. Elec. Eng. Comput. Sci. 2023 Jan;29:598-608.

[41] Miikkulainen R, Liang J, Meyerson E, Rawal A, Fink D, Francon O, Raju B, Shahrzad H, Navruzyan A, Duffy N, Hodjat B. Evolving deep neural networks. InArtificial intelligence in the age of neural networks and brain computing 2019 Jan 1 (pp. 293-312). Academic Press.

[42] Mijwil M, Salem IE, Ismaeel MM. The Significance of Machine Learning and Deep Learning Techniques in Cybersecurity: A Comprehensive Review. Iraqi Journal For Computer Science and Mathematics. 2023 Jan 7;4(1):87-101.

[43] Mijwil M, Unogwu OJ, Filali Y, Bala I, Al-Shahwani H. Exploring the Top Five Evolving Threats in Cybersecurity: An In-Depth Overview. Mesopotamian journal of cybersecurity. 2023 Mar 6;2023:57-63..

[44] Mohammadzad M, Karimpour J. Using rootkits hiding techniques to conceal honeypot functionality. Journal of Network and Computer Applications. 2023 May 1;214:103606.

[45] Neprash, Hannah T., Claire C. McGlave, Dori A. Cross, Beth A. Virnig, Michael A. Puskarich, Jared D. Huling, Alan Z. Rozenshtein, and Sayeh S. Nikpay. "Trends in ransomware attacks on US hospitals, clinics, and other health care delivery organizations, 2016-2021." In *JAMA Health Forum*, vol. 3, no. 12, pp. e224873-e224873. American Medical Association, 2022.

[46] Novikova J, Dušek O, Rieser V. The E2E dataset: New challenges for end-to-end generation. arXiv preprint arXiv:1706.09254. 2017 Jun 28.

[47] Pai V, Rao AS, Devidas, Prapthi B. An Intelligent Behavior-Based System to Recognize and Detect the Malware Variants Based on Their Characteristics Using Machine Learning Techniques. InInternational Conference on Advanced Network Technologies and Intelligent Computing 2022 Dec 22 (pp. 73-88). Cham: Springer Nature Switzerland.

[48] Pan, X., Yamaguchi, S., Kageyama, T., & Kamilin, M. H. B. (2022). Machine-learning-based white-hat worm launcher in botnet defense system. *International Journal of Software Science and Computational Intelligence (IJSSCI)*, *14*(1), 1-14.

[49] Qabalin MK, Naser M, Alkasassbeh M. Android spyware detection using machine learning: a novel dataset. Sensors. 2022 Aug 2;22(15):5765.

[50] Ozdamli, F., Ababneh, M., Karagozlu, D., & Aljarrah, A. (2022). Development and Testing of Performance Scale Application as an Effective Electronic Tool to Enhance Students' Academic Achievements. Electronics, 11(23), 4023.

[51] Revay G. An Overview of the Increasing Wiper Malware Threat. Threat Research. FortiGuard Labs. April 28, 2022.

[52] Sallow AB, Sadeeq M, Zebari RR, Abdulrazzaq MB, Mahmood MR, Shukur HM, Haji LM. An investigation for mobile malware behavioral and detection techniques based on android platform. IOSR Journal of Computer Engineering (IOSR-JCE). 2020;22(4):14-20.

[53] Sarker IH. Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective. SN Computer Science. 2021 May;2(3):154.

[54] Shaikh FA, Siponen M. Information security risk assessments following cybersecurity breaches: The mediating role of top management attention to cybersecurity. Computers & Security. 2023 Jan 1;124:102974..

[55] Sharma A, Malacaria P, Khouzani MH. Malware detection using 1-dimensional convolutional neural networks. In2019 IEEE European symposium on security and privacy workshops (EuroS&PW) 2019 Jun 17 (pp. 247-256). IEEE.

[56] Singh J, Singh J. Malware classification using multi-layer perceptron model. InInternational Conference on Innovative Computing and Communications: Proceedings of ICICC 2020, Volume 2 2021 (pp. 155-168). Springer Singapore.

[57] Singh P, Borgohain SK, Sarkar AK, Kumar J, Sharma LD. Feed-forward deep neural network (FFDNN)-based deep features for static malware detection. International Journal of Intelligent Systems. 2023 Feb 20;2023.

[58] Solanke AV, Patnaik GK. Intrusion detection using deep learning approach with different optimization. International Journal for Research in Applied Science and Engineering

Technology. 2020;8(5):128-34.
[59] Tahir R. A study on malware and malware detection techniques. International Journal of Education and Management Engineering. 2018 Mar 1;8(2):20.
[60] Thakkar A, Lohiya R. Fusion of statistical importance for feature selection in Deep Neural Network-based Intrusion Detection System. Information Fusion. 2023 Feb 1;90:353-63.
[61] Tuor A, Kaplan S, Hutchinson B, Nichols N, Robinson S. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. arXiv preprint arXiv:1710.00811. 2017 Oct 2.
[62] Varlioglu S, Elsayed N, ElSayed Z, Ozer M. The dangerous combo: Fileless malware and cryptojacking. SoutheastCon 2022. 2022 Mar 26:125-32.
[63] Velasco L, Signorelli M, De Dios OG, Papagianni C, Bifulco R, Olmos JJ, Pryor S, Carrozzo G, Schulz-Zander J, Bennis M, Martinez R. End-to-end intent-based networking. IEEE communications Magazine. 2021 Oct;59(10):106-12.
[64] Wang W, Zhao M, Wang J. Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network. Journal of Ambient Intelligence and Humanized Computing. 2019 Aug 1;10:3035-43.
[65] Wang Z, Han X, Kong W, Piao Y, Hou G, Watanabe M, Fukuda A. A Multi-Strategy Combination Framework for Android Malware Detection Based on Various Features. In2020 International Symposium on Theoretical Aspects of Software Engineering (TASE) 2020 Dec 11 (pp. 193-200). IEEE.
[66] Wichmann FA, Geirhos R. Are Deep Neural Networks Adequate Behavioral Models of Human Visual Perception?. Annual Review of Vision Science. 2023 May 26;9.
[67] Wilberforce T, Alaswad A, Garcia–Perez A, Xu Y, Ma X, Panchev C. Remaining useful life prediction for proton exchange membrane fuel cells using combined convolutional neural network and recurrent neural network. International Journal of Hydrogen Energy. 2023 Jan 1;48(1):291-303.
[68] Wu H. Mask classification using deep learning methods. InInternational Conference on Cloud Computing, Performance Computing, and Deep Learning (CCPCDL 2022) 2022 Oct 13 (Vol. 12287, pp. 504-509). SPIE.
[69] Xin Y, Kong L, Liu Z, Chen Y, Li Y, Zhu H, Gao M, Hou H, Wang C. Machine learning and deep learning methods for cybersecurity. Ieee access. 2018 May 15;6:35365-81.
[70] Yadav CS, Singh J, Yadav A, Pattanayak HS, Kumar R, Khan AA, Haq MA, Alhussen A, Alharby S. Malware analysis in iot & android systems with defensive mechanism. Electronics. 2022 Jul 28;11(15):2354.
[71] Yan S, Ren J, Wang W, Sun L, Zhang W, Yu Q. A Survey of Adversarial Attack and Defense Methods for Malware Classification in Cyber Security. IEEE Communications Surveys & Tutorials. 2022 Nov 28..
[72] Zhang T, Fu Q, Wang H, Liu F, Wang H, Han L. Bagging-based machine learning algorithms for landslide susceptibility modeling. Natural hazards. 2022 Jan;110(2):823-46.
[73] Zhang T, Quevedo RP, Wang H, Fu Q, Luo D, Wang T, de Oliveira GG, Guasselli LA, Renno CD. Improved tree-based machine learning algorithms combining with bagging strategy for landslide susceptibility modeling. Arabian Journal of Geosciences. 2022 Jan;15(2):183.
[74] Zhong Z, Gao Y, Zheng Y, Zheng B, Sato I. Real-world video deblurring: A benchmark dataset and an efficient recurrent neural network. International Journal of Computer Vision. 2023 Jan;131(1):284-301.
[75] Zhu HJ, Li Y, Wang LM, Sheng VS. A Multi-Model Ensemble Learning Framework for Imbalanced Android Malware Detection. Expert Systems with Applications. 2023 Jul 20:120952.