Open Access Article

MALWARE RECOGNITION SCHEME FOR ANDROID PLATFORM USING AN OPTIMIZED CNN-BASED APPROACH

Karrar Ahmed Kareem, Ethar Sabah Mohammad Ali
Ministry of Oil, Thiqar Oil Company, Iraq
College Of Engineering, Wasit University, Ministry Of Higher Education and Scientific Research, Iraq
kalknany526@gmail.com

Abstract
Malware is a malicious code which is developed to harm a computer or network. The number of malwares is growing so fast and this amount of growth makes the computer security researchers invent new methods to protect computers and networks. It is a very serious problem and many efforts are devoted to malware detection in today's cybersecurity world. In this research, a new method will be proposed to malware detection scheme for Android platform using an Optimized Convolutional Neural Networks based approach, which integrates both risky permission combinations and vulnerable API calls and use them as features in the CNN algorithm.

Keywords: Malware Detection, Deep Learning, Machine Learning, Security, Classification, Convolutional Neural Networks, Particle Swarm Optimization (PSO).

抽象的
恶意软件是一种恶意代码，旨在损害计算机或网络。 恶意软件的数量增长如此之快，这种增长量使计算机安全研究人员发明了保护计算机和网络的新方法。 这是一个非常严重的问题，在当今的网络安全世界中，许多努力都致力于恶意软件检测。 在这项研究中，将提出一种基于优化卷积神经网络的Android平台恶意软件检测方案的新方法，该方法集成了风险权限组合和易受攻击的API调用，并将它们用作CNN算法中的特征。

关键词：恶意软件检测、深度学习、机器学习、安全、分类、卷积神经网络、粒子群优化(PSO)。

Introduction

Malware, which is slipped into a victim's computer by hackers (attackers) through security vulnerabilities of the operating system or application software, can influence normal operation, collect sensitive information and steal superuser privileges in order to perform malicious actions. Generally, mainstream malware includes malicious scripts, vulnerability exploits, back doors, worms, trojans, spywares, rootkits, etc., and combinations or variations of the above types as well [1]. Annual reports from antivirus companies show that thousands of new malware are created every single day. This new malware become more sophisticated that they could no longer be detected by the traditional detection techniques such as signature-based Signature-based detection searches for specified bytes sequences into an object so that it can identify exception- ally a particular type of a malware. Its drawback is that it cannot detect zero-day or new malware since these mal ware

Received: October 18, 2021 / Revised: November 09, 2021 / Accepted: December 30, 2021 / Published: January 31, 2022
About the authors：Karrar Ahmed Kareem
Corresponding author- Email: kalknany526@gmail.com

signatures are not supposed to be listed into the signature database [2].

Identifying malware is essentially a software classification problem. The process of analyzing and classifying software involves two main steps. First, through program analysis, various descriptions of a program are generated and features are extracted from the program descriptions. Then classifiers can be constructed to perform classification based on these features. Due to the sheer number of apps that need to be processed, automated software analysis and classification is necessary. It is desired that little or no human effort be required in either of the steps [3]. To identify malware, most antivirus scanners use a combination of signature matching and heuristic-based detection [4]. The obvious problem with this is that only known and partially known samples based on the artifacts extracted by malware analysis will be recognized. Because of the rapid growth in malware samples over the last years, this is no longer sufficient.

Deep Learning is called deep learning as it is a kind of Machine Learning that imitates the human brain. Deep Learning, which means teaching a computer how to think, has moved away from research topics due to the problem of taking too much time to learn data. However, since AlexNet, who used CNN, a type of Deep Learning, won the 2012 image recognition algorithm competition, with 16.4 percent error rate, Deep Learning has drawn keen attention again [5]. Deep learning mechanisms itself facilitated to extract features by taking raw data set as input. They are mainly categorized into two types (1) convolution neural network (CNN) (2) recurrent neural network (RNN) [6].

In this proposal, we proposed a malware detection scheme for Android platform using an Optimized Convolutional Neural Networks based approach, which integrates both risky permission combinations and vulnerable API calls and use them as features in the CNN algorithm.

Main Challenges

- ✓ The most widely used method for malware detection is the signature-based method. The signature is sensitive to slight changes in malicious code. They require prior knowledge of malware samples. Signature-based approaches cannot detect modified or previously unseen malware. Therefore, one primary problem faced by the antivirus community is how to detect previously unseen malicious code.
- ✓ In the field of malware detection, collecting data samples is a relatively easy task; however, determining whether an executable is malicious or is not labor intensive. Thus, it is difficult for us to obtain a large amount of labeled data.
- ✓ How to build a malware detection system based on PSO-CNN.
- ✓ Whether the unlabeled data can be used to improve the accuracy of malware detection.
- ✓ Whether the deep representation generated using PSO-CNN is helpful for feature extraction and dimension reduction.

Purposes

- ✓ The goal of using PSO-CNN as a classifier is to reduce the dimensions of the feature vector and malware detection.

1-4 Summarization

It can be said that the main purpose of this research is proposing an optimizal approach for proposing a method name optimized CNN with PSO (PSO-CNN) for malware detection in Android platform. The rest of this research is organized as below:

- ✓ In chapter 2, will be consider some articles for liteture review.
- ✓ In chapter 3, the proposed method will be describe.
- ✓ In chapter 4, a simulation will be done and analysis methods in comparison to recent models.
- ✓ In chapter 5, a conclusion will be proposed and some methods which can used for futures to optimizing some remained challenges of this study will be consider.

Google's Android operating system is expected to increase dramatically with the 1,500 Android devices released by 2021. The company is currently leading the mobile operating system market with over 80% market share compared to iOS, Windows, Blackberry and Symbian. The availability of diverse Android markets such as Google Play, the official store, and third-party markets make Android a popular target not only for legitimate developers, but also for malware developers. More than one billion devices have been sold and over 65 billion downloads from Google Play. Android apps can be viewed in various areas, such as educational apps, game apps, social media apps, entertainment apps, banking apps, and more [7].

As an open source technology and widely used by users and developers, Android faces many challenges, especially in the case of malicious software. Malware-infected applications can send text messages to insurnce or bank numbers without user approval, access to personal data or even install code that can download and execute additional malware on the victim's device. The malware can also be used to create botnets. Over the past few years, the number of instances of malware attacking Android has increased significantly. According to a recent McAfee Inc. report, more than 2.5 million new Android malware programs were discovered in 2017, increasing the number of malware instances in various regions to approximately 25 million in 2017 [7].

Google in February 2012 introduced an identification mechanism to its app markets called Bouncer to reduce the prevalence of malware, including malware. The Bouncer tests presented their programs in a sandbox for 5 minutes to identify any harmful behaviors. However, it has been shown that Bouncer can be prevented by simple detection methods. At the same time as Bouncer, google introduced Google Play Protect and introduced it at the Google 2017 event. Google Play Protect is a regular service that automatically scans applications even after installation to ensure that installed applications are safe. It is reported that more than 50 billion programs are scanned and verified every day, no matter where they are downloaded. However, according to McAfee Inc. reporst, Google Play Protect also failed in testing against malware discovered in the last 90 days in 2017. In addition, most third-party stores are not capable of scanning and detecting harmful programs. Obviously there is still a need for additional research into effective ways to identify Android malware in the world in order to overcome the aforementioned challenges [7].

Various methods have been proposed in previous work aimed at detecting Android

malware. These approaches are classified into static analysis, dynamic analysis, or hybrid analysis (where static and dynamic are used together). Static analysis methods use reverse engineering to analyze malicious code. This chapter provides an overview of the issues in the field of malware and their use in the real world, along with the challenges involved in detecting them. It also discusses a series of previous research that has been done to detect malware on Android or other operating systems.

2-1 Malwares

Or malware is the common name for a number of types of malware, including viruses, ransomware, and spyware. Malware usually includes code generated by cyberattacks designed to cause extensive damage to data and systems or unauthorized access to a network. Malware is typically delivered as a link or file via email, requiring the user to click on the link or open the file to run the malware [100].

Malware has been under threat from individuals and organizations since the early 1970s when the Creeper virus first appeared. Since then, the world has been attacked by hundreds of thousands of different types of malware, all aiming to cause as much disruption and damage as possible. Malware delivers its load in several different ways. From ransacking to stealing sensitive personal information, cybercriminals are becoming more sophisticated in their methods. Below is a list of some of the common types and definitions of malware [8].

2-1-1 Types of Malware:

- ✓ Virus: Possibly the most common type of malware, viruses attach their malicious code to clean code and wait for an unsuspecting user or an automated process to execute them. Like a biological virus, they can spread quickly and widely, causing damage to the core functionality of systems, corrupting files and locking users out of their computers. They are usually contained within an executable file.
- ✓ Worms: Worms get their name from the way they infect systems. Starting from one infected machine, they weave their way through the network, connecting to consecutive machines in order to continue the spread of infection. This type of malware can infect entire networks of devices very quickly.
- ✓ Spyware: Spyware, as its name suggests, is designed to spy on what a user is doing. Hiding in the background on a computer, this type of malware will collect information without the user knowing, such as credit card details, passwords and other sensitive information.
- ✓ Trojans: Just like Greek soldiers hid in a giant horse to deliver their attack, this type of malware hides within or disguises itself as legitimate software. Acting discretely, it will breach security by creating backdoors that give other malware variants easy access.
- ✓ Ransomware: Also known as scareware, ransomware comes with a heavy price. Able to lockdown networks and lock out users until a ransom is paid, ransomware has targeted some of the biggest organizations in the world today — with expensive results.

2-1-2 How Does Malware Spread?

Each type of malware has its own unique way of causing destruction and relies heavily on the

type of user. Some strains are delivered via email or executable link. Others are delivered via instant message or social media. Even mobile phones are under attack. Organizations need to be aware of all vulnerabilities in order to determine an effective defensive line [8].

2-2 Deep Learning Review

Recently, there has been a lot of talk and discussion about all the possibilities of learning machines doing what people are currently doing in factories, warehouses, offices and homes. While this technology is evolving rapidly with fear and excitement, some terms such as artificial intelligence, machine learning and deep learning can make people anxious.

The field of artificial intelligence is basically when machines can perform tasks that usually require human intelligence. This includes machine learning in which machines can learn with experience and acquire their skills without human involvement. Deep learning is a subset of machine learning in which artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. Similarly to how we learn from experience, the deep learning algorithm does the same thing repeatedly, each time improving it slightly to optimize the result. Deep learning models benefit from more sophisticated artificial neural networks, more sophisticated system features and architecture, and can perform better than traditional machine learning models [14].

Deep learning allows machines to solve complex problems even when using a highly diverse, unstructured, interconnected dataset. The deeper the learning algorithms, the better they perform. Practically, deep learning is just a subset of machine learning. In fact, deep learning is technically a machine learning and works in a similar way. However, its capabilities are different. While the basic models of machine learning are getting better with respect to their performance, they still need some guidance. If an artificial intelligence algorithm returns a false prediction, then an engineer must step in and make adjustments. With the deep learning algorithm, an algorithm alone can determine whether the prediction is correct through its neural network [15].

A deep learning model is designed to continuously analyze data with a logical structure similar to how a human concludes. To achieve this, deep learning programs use a layered structure of algorithms called artificial neural networks. The design of an artificial neural network draws inspiration from the biological neural network of the human brain and results in a learning process that is far more powerful than standard machine learning models. It is certain that a deep learning model does not produce the wrong result as with other examples of artificial intelligence, it requires a lot of training to correct learning processes. But when it works as intended, deep functional learning is often perceived as a scientific surprise, with many finding it the backbone of real artificial intelligence [14].

An excellent example of deep learning is Google's AlphaGo. Google has developed a computer program with its own neural network that has learned to play the abstract board game called Go that requires sharp intellect and intuition. Playing against professional Go players, AlphaGo's deep learning paradigm learned how to play at a level never before seen in artificial intelligence, without telling when to make a specific move as a learning model. It saddened AlphaGo to defeat several "celebrity" masters of the game - not only could a machine

understand complex techniques and abstract aspects of the game, but also become one of its greatest players. Several advances are now in deep learning [15]:

- ✓ Algorithmic advances have enhanced the performance of deep learning methods.
- ✓ New approaches to machine learning have improved the accuracy of models.
- ✓ New classes of neural networks have been developed that are well suited for applications such as text translation and image classification.
- ✓ Much more data is needed to build neural networks with deep layers, including IoT data, social media text data, physician notes, and research transcripts.
- ✓ The computational advances of distributed cloud computing and graphics processing units have provided incredible computing power. This level of computational power is needed to train deep algorithms.

At the same time, human-machine interfaces have also evolved a lot. The mouse and keyboard are replaced by natural gesture, touch, and language, with a particular interest in artificial intelligence and deep learning. To solve deep learning problems due to the duplicate nature of deep learning algorithms, their complexity requires increasing computational power with the increasing number of layers and large amounts of data needed to train networks. The dynamic nature of deep learning methods - their ability to continually improve and adapt to changes in contextual information patterns - is a great opportunity to introduce more dynamic behavior in analytics. Further customizing customer analytics is a possibility. Another great opportunity is to improve the accuracy and performance of applications that have long used neural networks. Better depth can be added through better algorithms and greater computing power. While the current market focus is on deep learning techniques in cognitive computing applications, there is also great potential in more traditional applications of analysis, for example, time series analysis. Another possibility is to simply be more efficient and streamlined in analytical operations [15].

From an outside perspective, deep learning may be at the research stage, as computer scientists and data scientists are still testing its capabilities. However, deep learning has many practical applications that businesses are using today and others that are being used as research continues. A traditional approach is to analyze the use of data needed for engineering to extract new variables, then select an analytical model and finally estimate the parameters (or unknowns) of that model. These techniques can have predictive systems that are not well generalized because their completeness and accuracy depend on the quality of the model and its features. For example, if a fraud model is developed with attribute engineering, it starts with a set of variables and is most likely a model of those variables using data conversion. It may end up with the 30,000 variables that the model depends on, then the model needs to be shaped and understood which variables are meaningful, which are not, and so on. Adding more information will force people to redo. The new approach is deep learning, replacing formulation and model specifications with hierarchical properties (or layers) that learn to distinguish the underlying features of the data from the perspectives within the layers. Changing the paradigm by deep learning is a move from feature engineering to feature representation. The promise of deep learning is that it can lead to

predictive systems that are well-generalized, well-adapted, continually improving with new data entry, and are more dynamic than predicted systems. Hard business rules have been made [15].

2-3 Proposed Method Modeling

In this section, we first introduce the overall architecture of proposed malware detection scheme, and then describe each of the functions in details to demonstrate how the proposed scheme works for malware detection. Figure (3-1) depicts the overall structure of the malware detection scheme.

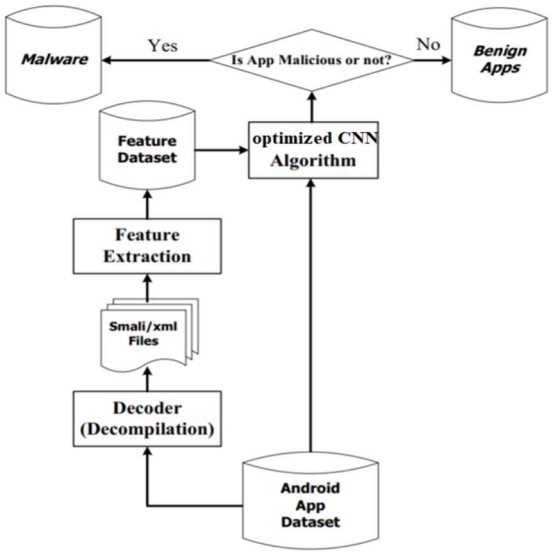

Figure (3-1) The schematic diagram of optimized CNN-based malware detection

There are three major components in the malware detection scheme, namely decoder (decompilation), feature extraction, and classifier. In the decompilation component, each Android app is unpacked and decoded into a readable smali file. Some key features, such as risky permissions, suspicious API calls, and URLs are then extracted in feature extraction components according to several important and widely accepted measures, such as TF-IDF and cosine similarity. Finally, we use Deep learning algorithm to build a classification model and evaluate them on the Android app dataset by classifying them into malware or benign apps.

Among several DL variations, Convolutional Neural Networks (CNNs) currently represent the state-of-the-art for complex classification problems, especially related to image analysis [1]. For the complex task at hand, we propose an PSO-optimized CNN classifier. The Traditional CNN features 3 layers, with convolution and max pooling. In specific case, the input tensor is a single vector with the features of selected the dataset. Each convolution layer i can be described by 3 hyper-parameters wi, wdi, hi, respectively; output channels, convolution window, and max pooling window. In addition to the convolution layers, there's a fully connected rectifier layer with a hyper-parameter w4 for the output channels, with a final softmax layer with dropout. Given the hyper-parameters, our objective becomes finding their optimal (or near-optimal) values, for our target classification problem. In order to reach this goal, we resort to PSO-based optimization. In fact, PSO algorithm will be used as feature extraction in hidden layers of CNN for tuning this neural network.

The first step is to enter the data into the CNN structure for training and testing with feature extraction operations, which uses PSO algorithms to construct a PSO-CNN structure to detect malware on Android operating systems. In fact, here's a modeling. A training set is constructed in $N^2 \times M$ dimensions where M is the number of data samples containing malware-free and malware-free data. The average data set is calculated by the equation (3-1).

$$\psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i$$

(1-3)

According to Equation (3-1), ψ is the mean data, M is the number of data and Γ_i is the data vector. The components corresponding to the highest eigenvalue are retained. These components define the malware space. Specific space is created by plotting data into the malware space, which creates components. So the weight vectors are calculated. Data dimensions are set to view specifications and data is refined in the pre-processing step of identification. Then the weight data vector and the malware weight vector are compared in the database. The average malware is calculated from the data and then subtracted from any data in the training set. Matrix A is constructed using the results of subtraction operations. The difference between each data and its mean is calculated as an equation (2–3).

$$\phi_i = \Gamma_i - \Psi\,, \qquad i = 1,2,\dots,M$$

(2-3)

According to equation (3-2), ϕ_i is the difference between the original data and the mean data. The matrix obtained by subtraction operation, which is matrix A, is multiplied by its derivation and eventually the covariance matrix C is formed which is the relation of the equation (3-3).

$$C = A^T A$$

(3-3)

According to equation (3-3), this A is formed by the difference between the vectors, for example $A = [\phi_1, \phi_2, \phi_3, \dots \phi_M]$. The dimension of matrix C is $N \times N$. The number of data samples or M is used to form the C matrix. In practice we can say that the matrix C is $N \times M$. On the other hand, when the order A is equal to M, only M of N is a special vector number of non-zero values. Then the eigenvalues of the covariance matrix are calculated. The components are then constructed by subtracting data from the number of special vector classes. The number of special vector classes means all the malware or malware in the data. The selected set of special vectors is multiplied by matrix A to reduce the feature space. Special vectors from smaller eigenvalues correspond to smaller changes in the covariance matrix. Other features of the data are maintained. The number of special vectors depends on the accuracy of the data in the database. A group of specially selected vectors are called components. Once the components are obtained, the data is collected in a database containing component space and the weights of the components in the same projected space. Special coefficients are compared to determine a data item with a special coefficient of database data. The component is then built into the data. The Euclidean distance between the data component and the components collected in the previous step is calculated. Malware is identified as an object whose Euclidean distance is less than the threshold value in the component database. If all Euclidean gaps are greater than the threshold, then malware will not be detected in the data and the data will be ignored. For the final application of the grid, it is necessary to identify the twists that there are three general methods for doing this, including thresholding of wavelet coefficients, adaptive filters, and thresholding of the range of action potentials. The approach of this research to detect malware is to use the threshold of action potentials. The value of this threshold is determined by the equation (3-4).

$$\begin{cases} \sigma_n = median\{\dfrac{|x|}{0.6745} \\ Threshold = 3.5\ \sigma_n \end{cases}$$

(4-3)

In relation (4–4), the x signal or malware data recorded by the microelectrode (raw signal) and σ_n is an estimate of the standard deviation of the

noise. It is important to note that if we use the standard deviation of the signal, a larger threshold value will be obtained and as a result some twists will be mistakenly eliminated. After selecting the threshold value, the torsions are aligned according to their maximum values. The precise alignment of the twists is a crucial factor in identifying malware with twists. Like other neural networks, this neural network also requires training, with the goal of finding a mapping such as $f: R^n \to R$ in equation to (3-5).

$$f(v) = \sum_i^n w_i \varphi(||v - C_i||) \tag{5-3}$$

According to Equation (3-5), $v \in R^n$ is a 32-point vector for the input and the Gaussian basic function $\varphi(0)$ is determined by Equation (3-6).

$$\varphi(v) = exp(\frac{-v^2}{2\sigma^2}) \tag{6-3}$$

Then, it is necessary to calculate the error corresponding to each sample from the gradient decsent and for the random initial values for the weights, for each training sample, in relation to (3-7).

$$e_i = t_i - y_i = t_i \sum_{j-1}^{N} w_j \varphi(||v_i - C_j||) \tag{7-3}$$

Therefore, the total error of the network for all training input vectors or P of the data is $E = \frac{1}{2}\sum_{i=1}^{P}|e_i|^2$. If the errorE is less than the threshold error, the training ends. This value is set manually at the beginning of the job. Otherwise weights will be updated using the gradient gradient. Upon completion of training with the neural network, the ability of each twist to belong to the class to which it belongs is achieved. It needs to be characterized by its layering structure, which includes the input layer (neurons), the secret layer where the training operation is carried out, which includes the three inner layers of twisting, pooling and fully connected, as well as the final test operation on the face output layer. The issue of detecting the presence or absence of malware creates a challenge and a search space. In an optimization problem such as malware detection, the presence or absence of malware from the data with the next N_{var} would be an x_{Nvar} array that represents the current position for the coiled layer in the convolutional neural network. This array is defined as equation (3-8).

$$Convolve = [x_1, x_2, \dots, x_{Nvar}] \tag{8-3}$$

The degree of appropriateness (or amount of benefit) in the current Convolve layer is obtained by evaluating the malware function (f_p) in Convolve. So there are equations (3-9) and (3-10).

$$profit = f_p(Convolve) = f_p(x_1, x_2, \dots, x_{Nvar}) \tag{9-3}$$

$$j\left(x^{(i)}, \dots, x^{(n)}, \theta^{(1)}, \dots, \theta^{(n)}\right) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2 \tag{10-3}$$

In fact, the overall purpose of the malware detection function is whether or not it is related (3-10). In general, we should minimize the above function as much as possible to detect malware. In fact, a modal deletion of additional components is considered for precise detection of malware that minimizes the equation (11–3).

$$\min_{\substack{x^{(1)},\dots,x^{(n_m)} \\ \theta^{(1)},\dots,\theta^{(n_u)}}} j\left(x^{(1)},\dots,x^{(n)},\theta^{(1)},\dots,\theta^{(n)}\right) \quad (11\text{-}3)$$

As can be seen, the deep neural network structure used is a convolution algorithm that maximizes the malware detection function or not. To use a deep CNN to solve minimization problems such as malware detection, it is sufficient to multiply a negative sign in the cost function. To start the convolution optimization algorithm, a $Convolve$ matrix is generated with $N_{pop} * N_{var}$. Then, for each of these Convolves, a random number of Layers is assigned. Pooling layers are generally between 2 and 5 items. These numbers are used as the upper and lower limits of the Poolling assignment to each convolv in depth training in different iterations. Another custom of any deep structure based CNN is that they have layers in a given domain. Hence, the maximum amplitude of layers connected in the CNN is called $Max_{Connected\ Layer}$. In an optimization problem, the upper limit of the variables var_{high} and the lower limit var_{low} each layer will have a $Max_{Connected\ Layer}$ that is proportional to the total number of layers, the number of layers of current training data, and also The upper and lower limit are the variables of the problem. Therefore $Max_{Connected\ Layer}$ is defined as equation (3-12).

$$Max_{Connected\ Layer} = \alpha \times \frac{Number\ of\ current\ layers}{Total\ number\ of\ layers} \times \left(Var_{high} - Var_{low}\right) \quad (12\text{-}3)$$

In equation (3-12), α is the variable with which the maximum value of $Max_{Connected\ Layer}$ is set. In equation (3-12) there are also θ layers and in relation (3-11), λ is the estimator value. Each convolve segment in the deep CNN only λ % of all detected areas toward the current ideal target and also has a deviation of φ radians. These two parameters help each convolve part of the deep CNN convergence to search for more environment. λ is a random number between 0 and 1 and φ is a number between π/6 and -π/6. When all of convolve in the deep CNN migrated to the target point and new habitats were identified, each convolve part in the deep CNN had some layers. Depending on the number of layers in each convolve segment in the deep CNN, a $Max_{Connected\ Layer}$ is specified for it, and then communication with the layers begins. It is now necessary to optimize this deep CNN using the PSO algorithm for the extraction of correct and optimal features. It is assumed that the dataset is M which represents the number of training images, F_i average images, and L_i per T_i vector image. Initially M is the number of data that each data contains $M \times N$ dimension. Any data can be represented in N space by $A = N * N * M$. Therefore, it is necessary to provide a hybrid layer for the PSO algorithm segment with deep CNN, which will be in equation (13-3).

$$x_i^{(l+1)} = (w_d)_i^{(l+1)}(y_d)^{(l)} + (w_f)_i^{(l+1)}\left(y_f\right)^l + b_i^{(l+1)} \quad (13\text{-}3)$$

In equation (3-13), y_d is the output of the deep CNN and y_f is the output of the PSO algorithm that has two parts w_d and w_f for weight. In fact, w_d determines the weight of the deep CNN and w_f is the weight of the PSO algorithm. This layer is used as $\hat{y}$ to combine these two layers, namely the deep CNN and the PSO algorithm section, which has an activation function that is considered as a hyperbolic tangent. In fact, the classification and detection section was developed using an innovative method called the

deep CNN based on the PSO algorithm or PSO-CNN.

What separates the proposed approach from the neural-fuzzy network (ANFIS) structure is that in neural-fuzzy network structures, the definition of membership functions and fuzzy numbers is based on inputs and some properties especially in images whose features are And are connected to an inference system that is trained on a weighted and bias-based training layer with a transfer or stimulus function. But in the proposed approach structure that performs the detection operation, the structure of the PSO algorithm is located on the training layers or hidden layers of the deep CNN section after the twisting layer, where the twisting layer, the PSO algorithm layer. The optimum feature extraction is simultaneous with the instruction, the pooling layer and the fully connected layer, with the filter applied in a 3 × 3 window type. In fact, in this training operation, the coefficients between the PSO algorithm are applied with the layers of the convoluted segment, ie the convolve layer, and the output layer displays the classification based on the data obtained as a vector matrix and is processed definitely to prove the optimization problem is done on the input data itself.

2-4 Evaluation

Several evaluation criteria have been used in this study, including Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Signal-to-Noise Ratio (SNR), and precision criteria. At the beginning, each survey is completed and finally the results of the research performed with each evaluation criterion are mentioned.

3- 1 Mean Square Error (MSE)

In this project, the MSE are used to evaluate the system. The purpose of the evaluation here is to guarantee the proposed approach to other methods. It should be noted that the mean error squares also have a training stage that is not considered in this study and its training is used when using neural network structures and then the mean error squares as a Use the evaluation method. To identify the parameters, we need training data expressed as an equation (3-14).

$$\{(u_i; y_i), i = 1, \dots, m\} \tag{14-3}$$

Replacing the input and output pairs in the main equation, we arrive at the linear equation m such as euation(3-15).

$$\begin{cases} \{f_1(u_1)\theta_1 + f_2(u_1)\theta_2 + \cdots + f_n(u_1)\theta_n = y_1 \\ \{f_1(u_2)\theta_1 + f_2(u_2)\theta_2 + \cdots + f_n(u_2)\theta_n = y_2 \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad . \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad . \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad . \\ \{f_1(u_m)\theta_1 + f_2(u_m)\theta_2 + \cdots + f_n(u_m)\theta_n = y_m \end{cases} \tag{15-3}$$

Which is in the form of a relation matrix such as equation (3-16).

$$A\theta = Y \tag{16-3}$$

And matrix A is like equation (3-17).

$$A = \begin{bmatrix} f_1(u_1) & \cdots & f_n(u_1) \\ \vdots & \vdots & \vdots \\ f_1(u_m) & \cdots & f_n(u_m) \end{bmatrix} \tag{17-3}$$

And θ is a vector $n \times \theta$, which is the same as equation (3-18).

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} \tag{18-3}$$

And y is the output vector $m \times 1$ and such as equation (3-19).

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix}$$

(19-3)

In this case, the vector θ can be calculated as equation (3-20).

$$\theta = A^{-1}Y$$

(20-3)

However, the number of input-output data pairs is usually greater than the number of equations. In general, the data may be accompanied by noise that is too high or the model may not be able to accurately determine the output. In this case the equation is in the form of equation (3-21).

$$A\theta + e = Y$$

(21-3)

In this case we will seek $\theta = \hat{\theta}$ to minimize the sum of the squares of the error and be such as equation (3-22).

$$E(\theta) = \sum_{i=1}^{m}(y_i - a_i^T\theta)^2 = e^T e = (y - A\theta)^T(y - A\theta)$$

(22-3)

If AA^T is reversible then we have the equation (3-23).

$$\theta = (A^T A)^{-1}A^T y$$

(23-3)

In each iteration, the feedback method is forwarded until the matrix A and the outputs are obtained by the least squares error method. It should be noted that all training data must be applied and the basic parameters kept constant. Then the empty parameters are kept constant and the initial parameters are set by the descending gradient. When the minimum value indicates less erosive variance, the value is usually expressed as a percentage. Percentage value refers to the fraction of data we want to measure accuracy, sensitivity, and feature rates. Because the mean squared error can be either a variance or an integer below 5, the best rate is between 0.1 and 0.9. Based on the change in the mean squared error as the estimator, the results of other estimation methods may change and have a high impact on them. In fact, this same mean of error squares affects other estimation methods based on this result rate, a percentage for accuracy, sensitivity and specificity estimation methods, and a number in dB for the peak signal ratio. The noise and signal-to-noise ratio is obtained.

3- 2 Peak Signal-to-Noise Ratio (PSNR)

Another criterion used to evaluate is the Peak Signal-to-Noise Ratio (PSNR), which is the ratio between the maximum possible signal power and the noise power calculated from the equation (24-23) in dB.

$$PSNR = 10.log_{10}(\frac{MAX_I^2}{MSE})$$

(24-3)

Where MAX_I^2 is the maximum possible signal.

3- 3 Signal-to-Noise Ratio (SNR)

There is a need for a benchmark to represent the useful signal versus noise signal in the built project. A value less than 1 indicates a serious noise problem in the images. A value greater than 1 is a satisfactory condition and a value above 2 is appropriate. This claim can be substantiated by citing scientific articles as well as explanations available on most websites. In fact, the higher this indicator is, the better the signal is in the image. The Signal-to-Noise Ratio (SNR) is defined as the Signal-to-Noise Ratio of the power to noise ratio according to equation (3-25).

$$SNR = \frac{P_{signal}}{P_{noise}} \quad (25\text{-}3)$$

Where P is the average signal power value. Because most signals have a dynamic range, they are defined in decibel logarithms, which will be the equation (3-27) for the power signal and the equation (3-27) for the noise signal.

$$P_{signal,dB} = 10log_{10}(P_{signal}) \quad (26\text{-}3)$$

$$P_{noise,dB} = 10log_{10}(P_{noise} \quad (27\text{-}3)$$

3- 4 Precission (Accuracy)

The accuracy rate is a criterion expressed as a percentage, which is the most important overall result of the evaluation criteria section, which is the accuracy of the relationship and its eqution is (3-28).

$$Accuracy = 100 \times \frac{TP + TN}{TN + TP + FN + FP} \quad (28\text{-}3)$$

In equation (3-28), TP is false positive, TN positive negative, false positive FP and false negative FN. Equation (3-29) shows the sensitivity expressed in percentage.

$$Sensitivity = \frac{TP}{TP + FN} \quad (29\text{-}3)$$

Equation (3-30) shows the data properties expressed in percentage.

$$Specificity = \frac{TN}{TN + FP} \quad (30\text{-}3)$$

3- 5 ROC and AUC

This curve is known as one of the most important evaluation criteria that expresses the criteria for measuring the efficiency of a classification or clustering operation in a system. In general, the ROC curve is a graphical representation of the sensitivity or prediction accuracy versus the false prediction in a binary classification system in which the threshold of separation varies. The ROC curve is also shown by plotting the predicted positives against the predicted false positives. The area under the ROC curve is a number that measures one aspect of performance. This area below the curve is called the AUC, which is between zero and one. The value is 0.5 times the random prediction, and values above 0.7 to 1 indicate excellent prediction, classification or clustering.

Simulation and Results

4-2 Simulation and Results

MATLAB software will be used to implement the approach as well as using a set of evaluation criteria as the simulation platform method. In fact, there is a simulation and evaluation of the method, along with a Java-based structure for Android that can be used. The main purpose is to evaluate the proposed approach, called PSO-CNN, which can be compared with other methods in this field to determine the accuracy of malware detection.

Feature selection and extraction is one of the issues in machine learning as well as statistical pattern recognition. This is important in many applications because there are many features that many are either useless or have little information about. Not removing these features does not cause any information problems, but it does increase the computational burden for the intended application and in addition it saves a lot of useless information along with useful data.

For the feature selection problem, there are many solutions and algorithms, some of which are thirty or forty years old. Different feature

selection methods try to find the best one among the 2^N candidate subfolders. In all of these methods, based on the application and the type of definition, a subset is selected as the answer that can optimize the value of an evaluation function. Although every method tries to pick the best one, but given the breadth of possible solutions and how these solution sets increase with N, finding the optimal solution is difficult. And in medium to large N, it is very expensive. In this research, a method based on the PSO-CNN hybrid method is presented where the PSO algorithm is placed as a layer in the hidden layers phase of CNN. In fact, CNN's data training and feature extraction operations are to be done simultaneously in its PSO-based training layer. In this simulation, a set of measurement criteria is taken into account, and the conditions in the two reference papers [25] and [26] are also modeled and simulated in the same research method to compare them completely. Be. It is noteworthy that the proposed approach and two reference articles use a database called MS Big Malware Database, taken from the https://www.kaggle.com/c/malware-classification website. In general, the MS BIG dataset with malware class distribution and benign instances is shown in Table (4-1).

Table (4-1) MS BIG dataset with malware class distribution and benign instances

Malware Type	Number of Samples
Ramnit (R)	1534
Lollipop (L)	2470
Kelihos ver3 (K3)	2942
Vundo (V)	451
Simda (S)	41
Tracur (T)	685
Kelihos ver1 (K1)	386
Obfuscator.ACY (O)	1158
Gatak (G)	1011
Benign Sample (B)	1200

The general configurations of the Convolutional Neural Network (CNN) are discussed in detail in Chapter 3, but in the simulation, additional parameters are required for the Convolutional Neural Network (CNN) as shown in Figure (4-1). The network uses two layers for training, both using the trainlm or Levenberg Marquardt or trainlm activation function with 10 neurons. The number of neural network cycles is assumed to be 1000 rpm. The mutation rate is also 0.001. The weight of each layer is 1. The layers are named sequentially in Chapter 3, which consists of a hidden layer in two sections with convolve layers, an PSO algorithm layer for feature extraction, a pooling layer and a fully connected layer.

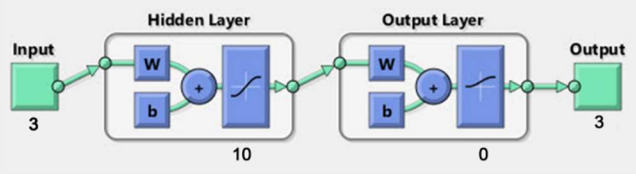

Figure (4-1) PSO-CNN in MATLAB configuration

It is then necessary to introduce the PSO algorithm operators as presented in Table (4-2) by default.

Table (4-2) PSO algorithm operator's rate

Initial Population (Particles)	100
Iteration Numbers	100
Velocity	2
F rate for operating mobility	0.0001
CR rate for operating mobility of two features (malware – benign)	0.1
Number of all operators	[2 12]
The number of strong agents per iteration	2
The number of eliminating factors per iteration	4

Count the best possible solution and analyze its best	6

Outputs are displayed after all the initial values are set. Initially, Figure (4-2) is shown as follows.

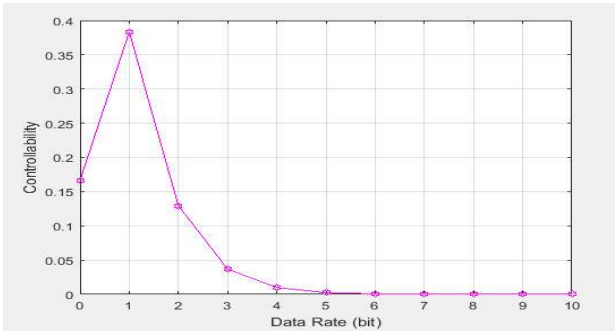

Figure (4-2) Bit Rate Control in Malware Detection after Proposed Method

As can be seen in Figure (4-2), bit rate data controllability in malware detection after applying the proposed method is initially increased due to the use of appropriate number of agents in the environment. It's gone downhill and it's actually been optimized. The controllability reflects the fact that the proposed approach is capable of controlling malware. Figure (4-3) shows the bitrate data for dimensionality change in malicious areas compared to the methods presented in [25] and [26].

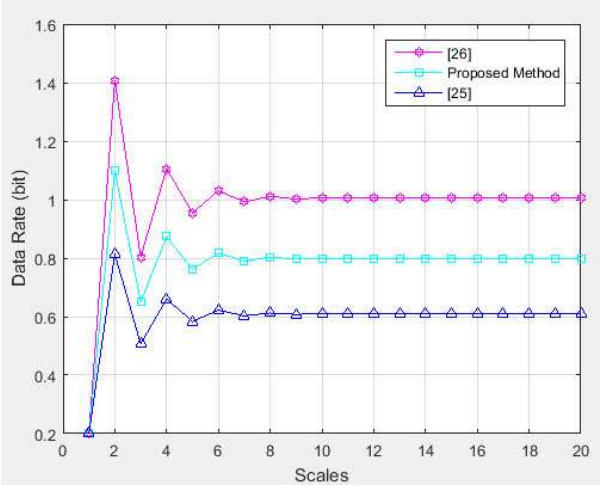

Figure (4-3) Bit rate data on dimensionality change in malicious areas

As shown in Figure (4-3), bit rate data on dimensionality change is shown in malware areas. In fact, it is clear that as the dimensions get larger, the bit rate data is more likely to be sent, and this is a common occurrence. But since the growth of data volume in bits may include malware, it is shown in scales. It can be seen that the proposed approach with cyan color diagrams has a functional advantage over the other two methods, namely [25] with blue and reference [26] with pink. Prior to testing the data by PSO-CNN, the accuracy was approximately 70%, which can be seen in Figure (4-4), whose y-axis is 100% unit.

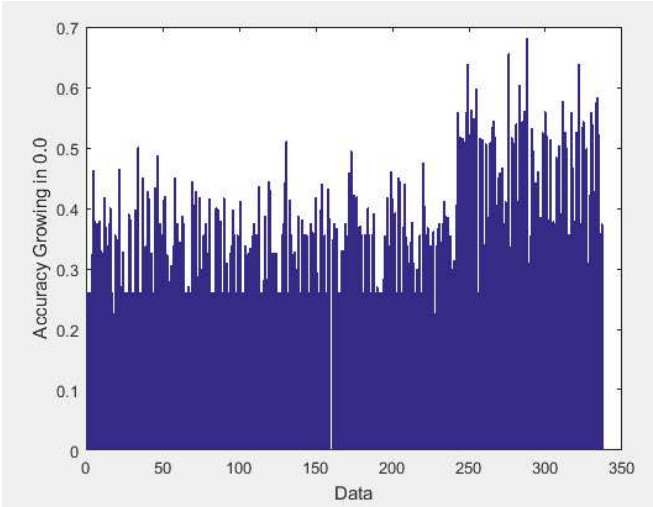

Figure (4-4) approximate accuracy before data test operation for detection and classification by PSO-CNN method

But after thorough analysis of the proposed approach, the accuracy of malware detection by the PSO-CNN method reaches 98.54%. In general, the results of the evaluation of the proposed approach are shown in Table (4-3) and the result of the ROC chart and the AUC rate in Figure (4-5).

Table (4-3) results of the evaluation of the proposed approach

AUC	Specificity (%)	Sensitivity (%)	Accuracy (%)	MSE
0.8592	85 %	90 %	98.54 %	0.45

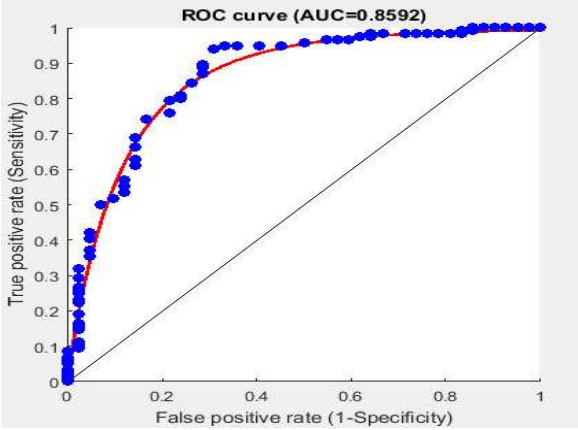

Figure (4-5) ROC and AUC rate after applying proposed method

The results of the proposed approach are found to be relatively appropriate. Also in Figure (4-5) the data that is out of line of the ROC and is disconnected from it are actually errors or outliers. The red line is the ROC line, where it rises and is calculated based on the positive integer or TP on the false integer or FP, the AUC rate or the area under its curve. The middle black line is the regression line. A comparison of the accuracy criteria is also made with the two methods presented in [25] and [26], the results of which can be seen in Table (4-4) and Figure (4-6). Also in these two references, there are a number of other methods that have been compared with their predecessor methods, which is also done in this study. It is noteworthy that the validation for accuracy was based on the Kfold method, with a K rate of 10 which would be 10Kfold.

Table (4-4) Evaluation results in comparison to recent methods based on accuracy

References	Approaches	Accuracy (%)
Zhongru Ren, et al., 2020 [25]	CRNN-LSTM	95.80 %
Xinbo Liu, et al., 2020 [26]	Visual-AT	97.56 %
Zhongru Ren, et al., 2020 [25] – other methods	CNN	93.60 %
Zhongru Ren, et al., 2020 [25] – other methods	CRNN-GRU	93.70 %
Xinbo Liu, et al., 2020 [26] - other methods	Random Forest	95.13 %
Xinbo Liu, et al., 2020 [26] - other methods	LZJD-KNN	96.78 %
Xinbo Liu, et al., 2020 [26] - other methods	M-CNN	98.05 %
Xinbo Liu, et al., 2020 [26] - other methods	Lanzcos-CNN	95.61 %
Proposed Method	PSO-CNN	98.54 %

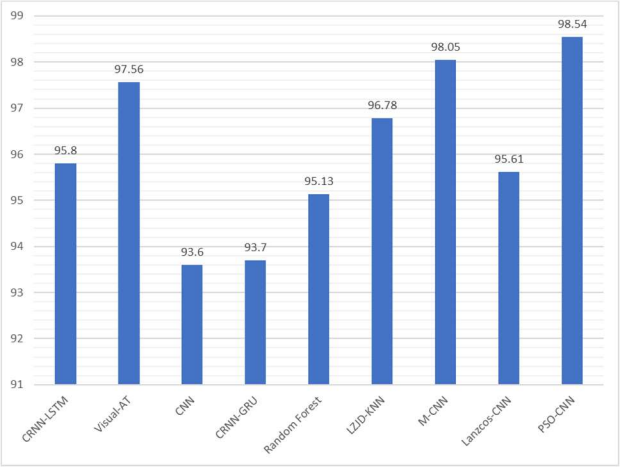

Figure (4-6) Evaluation results in comparison to recent methods based on accuracy (%)

It is found that the accuracy of the proposed approach in detecting malware is superior to previous methods when using the same MS Big data set. Following is a comparison of the area under the curve or the AUC with the method presented in [25], the results of which are presented in Table (4-5) and Figure (4-7).

Table (4-5) Evaluation results in comparison to recent methods based on AUC

References	Approaches	AUC
Zhongru Ren, et al., 2020 [25]	CRNN-LSTM	0.982
Zhongru Ren, et al., 2020 [25] – other methods	CNN	0.980
Zhongru Ren, et al., 2020 [25] – other methods	RNN-LSTM	0.975
Zhongru Ren, et al., 2020 [25] – other methods	RNN-GRU	0.983
Proposed Approach	PSO-CNN	0.8592

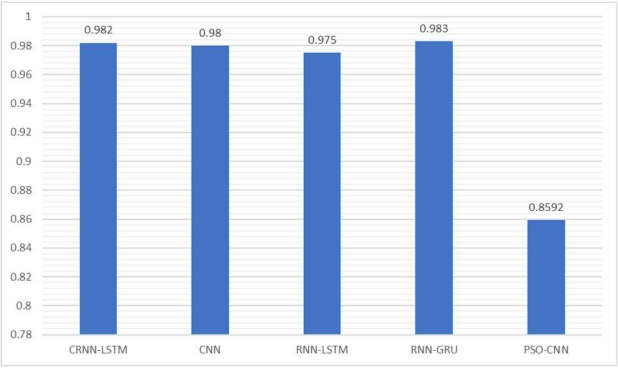

Figure (4-7) Evaluation results in comparison to recent methods based on AUC

It can be seen from Table (4-5) and Figure (4-7) that the area under the curve, or AUC, of the proposed approach, which should be numerically lower than a standard, is better than previous methods.

Conclusion and Suggestions

Conclusion

Nowadays, mobile devices such as smartphones and tablets have become very popular, due to a reduction in their cost and an increase in their functionalities and services availability. Moreover, the growing trend of implementing bring your own device (BYOD) policies in organizations has also contributed to the adoption of these technologies, not only for everyday communication activities but to support enterprise systems, industrial applications, and commercial transactions, which raise new security issues. In this scenario, operating systems have also played an important role in the adoption and proliferation of mobile devices and applications, giving also space for the appearance of malicious software (malware). This is the case for the Android OS, which, due to its openness and free availability, has become not only a major stakeholder in the market of mobile devices but has also become an attractive target for cybercriminals.

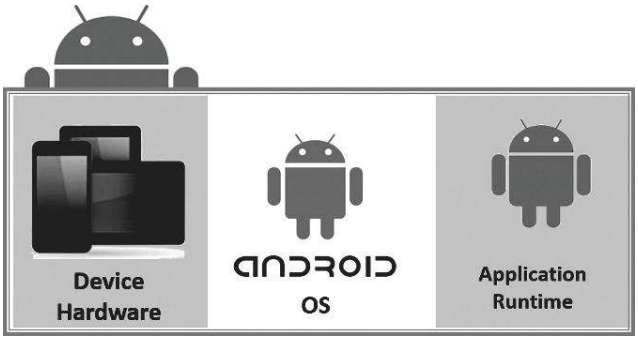

Figure (5-1) the main Android platform building blocks

First of all, the Android device hardware block refers to the wide range of hardware configurations where Android can be run, including smartphones, tablets, watches, automobiles, smart TVs, OTT gaming boxes, and set top boxes. Android is processor-agnostic, but it does take advantage of some hardware-specific security capabilities such as ARM eXecute-Never. Secondly, the Android OS building block refers to the Android OS itself, which is built on top of the Linux kernel, thus all device resources are accessed through the operating system. Thirdly, the Android application runtime block refers to the managed runtime used by applications and some system services on Android. In this case, it must be taken into account that applications are written in the Java language and run in the Android runtime (ART). However, many applications, including core Android services and applications, are native applications or included native libraries. Both ART and native applications run with the same security environment, contained with the applications sandbox. Thus, applications get a dedicated part of the file system in which they can write private data, including database and raw files.

Android malware can be characterized in different ways: a systematic characterization is proposed ranging from their installation, activation, to the carried malicious payloads. Thus, malware installation can be generalized into three main social engineering-based techniques: repackaging, update attack, and drive-by download. Repackaging is one of the most common techniques that malware authors use to piggyback malicious payloads into applications. In essence, malware authors get an application file, disassemble them, enclose malicious payloads, reassemble, and submit the new application to an official or alternative market. Users could be vulnerable by being enticed to download and install these infected applications. In the case of the update attack, instead of enclosing the payload as a whole only an update component is included, which will fetch or download the malicious payloads at runtime. Because the malicious payload is in the "updated" application, not the original application itself, it is stealthier than the malware installation technique that directly includes the entire malicious payload in the first place. The third technique applies the traditional drive-by download attack to mobile space. Though they are not directly exploiting mobile browser vulnerabilities, they are essentially enticing users to download "interesting" or "feature-rich" applications. This is only a set of common techniques, other threats include combinations of the previous techniques, as well as approaches such as "spyware," which intend to be installed to victim's phones on purpose; fake apps that masquerade as the legitimate applications but stealthily perform malicious actions, such as stealing users' credential; applications that provide the functionality they claimed, they are not fake ones, but that intentionally include malicious functionality, which is unknown to users.

Malware programs currently represent the most serious threat to computer information systems. Despite the performed efforts of researchers in this field, detection tools still have limitations for one main reason. Actually, malware developers usually use obfuscation techniques consisting in a set of transformations that make the code and/or its execution difficult to analyze by hindering both manual and automated inspections. These techniques allow

the malware to escape the detection tools, and hence to be seen as a benign program.

The approach presented by this study is to use a hybrid method of Convolutional Neural Network (CNN) and Particle Swarm Optimization (PSO) algorithm. In fact, the Convolutional Neural Network (CNN) is intended to train and test data to make a diagnosis, but it is important to identify malware features from a dataset, and the Convolutional Neural Network (CNN) cannot be precisely identifiy feature extraction, so it uses Particle Swarm Optimization (PSO) algorithm to improve it in the training and test layers. In fact, all the data is in the input layer and in the hidden layer, first is the convolve layer, then the Particle Swarm Optimization (PSO) algorithm layer, and then the pooling layers and finally the fully connected layer. The output layer is also intended for displaying malware work and detection. The results show that the accuracy of the proposed approach has a relative advantage over the previous methods and in terms of the AUC based on the ROC curve, optimization with the previous methods is also evident.

References

1. Ni, Sang, Quan Qian, and Rui Zhang. "Malware identification using visualization images and deep learning." Computers & Security 77 (2018): 871-885.
2. Kumar, Rajesh, Zhang Xiaosong, Riaz Ullah Khan, Ijaz Ahad, and Jay Kumar. "Malicious code detection based on image processing using deep learning." In Proceedings of the 2018 International Conference on Computing and Artificial Intelligence, pp. 81-85. ACM, 2018.
3. Nix, Robin, and Jian Zhang. "Classification of Android apps and malware using deep neural networks." In 2017 International joint conference on neural networks (IJCNN), pp. 1871-1878. IEEE, 2017.
4. Bragen, Simen Rune. "Malware detection through opcode sequence analysis using machine learning." Master's thesis, 2015.
5. Lee, Yoon-seon, Jae-ung Lee, and Woo-young Soh. "Trend of Malware Detection Using Deep Learning." In Proceedings of the 2nd International Conference on Education and Multimedia Technology, pp. 102-106. ACM, 2018.
6. Vinayakumar, R., K. P. Soman, and Prabaharan Poornachandran. "Detecting malicious domain names using deep learning approaches at scale." Journal of Intelligent & Fuzzy Systems 34, no. 3 (2018): 1355-1367.
7. Mohammed K. Alzaylaee, Suleiman Y. Yerima, and Sakir Sezer. DL-Droid: Deep learning based android malware detection using real devices. Computers & Security, Volume 89, 2020, 11 Pages.
8. S. Sibi Chakkaravarthy, D. Sangeetha, and V. Vaidehi. A Survey on malware analysis and mitigation techniques. Computer Science ReviewVolume 32May 2019Pages 1-23.
9. Sacha Gobeyn, Ans M. Mouton, Anna F. Cord, Andrea Kaim, and Peter L. M. Goethals. Evolutionary algorithms for species distribution modelling: A review in the context of machine learning. Ecological Modelling, Volume 39224 January 2019, Pages 179-195.

10. Maurice Clerc. Particle Swarm Optimization. First published:1 January 2006, 243 Pages.
11. Sara Kaviani, and Insoo Sohn. Influence of random topology in artificial neural networks: A survey. ICT Express, in press, corrected proof, Available online 14 February 2020.
12. Long Jin, Shuai Li, Bin Hu, and Mei Liu. A survey on projection neural networks and their applications. Applied Soft Computing, Volume 76, March 2019, pp. 533-544.
13. Martin T. Hagan. Neural network design. 1996-2017.
14. Geert Litjens, Francesco Ciompi, Jelmer M. Wolterink, Bob D. de Vos, and Ivana Išgum. State-of-the-Art Deep Learning in Cardiovascular Image Analysis. JACC: Cardiovascular Imaging, Volume 12, Issue 8, Part 1, August 2019, pp. 1549-1565.
15. Nikhil Buduma. Fundamentals of Deep Learning, Designing Next-Generation Machine Intelligence Algorithms. 298 Pages, 2017.
16. Philip G. Brodrick, Andrew B. Davies, and Gregory P. Asner. Uncovering Ecological Patterns with Convolutional Neural Networks. Trends in Ecology & Evolution, Volume 34, Issue 8August 2019, Pages 734-745.
17. Arshad, Saba, Munam A. Shah, Abdul Wahid, Amjad Mehmood, Houbing Song, and Hongnian Yu. "Samadroid: a novel 3-level hybrid malware detection model for android operating system." IEEE Access 6 (2018): 4321-4339.
18. Ma, Zhuo, Haoran Ge, Yang Liu, Meng Zhao, and Jianfeng Ma. "A Combination Method for Android Malware Detection Based on Control Flow Graphs and Machine Learning Algorithms." IEEE Access 7 (2019): 21235-21245.
19. Cui, Zhihua, Fei Xue, Xingjuan Cai, Yang Cao, Gai-ge Wang, and Jinjun Chen. "Detection of malicious code variants based on deep learning." IEEE Transactions on Industrial Informatics 14, no. 7 (2018): 3187-3196.
20. Li, Jin, Lichao Sun, Qiben Yan, Zhiqiang Li, Witawas Srisa-an, and Heng Ye. "Significant permission identification for machine-learning-based android malware detection." IEEE Transactions on Industrial Informatics 14, no. 7 (2018): 3216-3225.
21. Azmoodeh, Amin, Ali Dehghantanha, and Kim-Kwang Raymond Choo. "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning." IEEE Transactions on Sustainable Computing 4, no. 1 (2018): 88-95.
22. Kim, TaeGuen, BooJoong Kang, Mina Rho, Sakir Sezer, and Eul Gyu Im. "A multimodal deep learning method for Android malware detection using various features." IEEE Transactions on Information Forensics and Security 14, no. 3 (2018): 773-788.
23. Daniel Gibert, Carles Mateu, and Jordi Planes. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. Journal of Network and Computer Applications, Volume 1531 March 2020.
24. Ali Feizollah, Nor Badrul Anuar, Rosli Salleh, and Ainuddin Wahid Abdul

Wahab. A review on feature selection in mobile malware detection. Digital Investigation, Volume 13, June 2015, Pages 22-37.

25. Zhongru Ren, Haomin Wu, Qian Ning, Iftikhar Hussain, and Bingcai Chen. End-to-end malware detection for android IoT devices using deep learning. Ad Hoc Networks, Volume 101, 15 April 2020.
26. Xinbo Liu, Yaping Lin, He Li, and Jiliang Zhang. A novel method for malware detection on ML-based visualization technique. Computers & Security, Volume 8, 9 February 2020.
27. Manel Jerbi, Zaineb Chelly Dagdia, Slim Bechikh, and Lamjed Ben Said. On the use of artificial malicious patterns for android malware detection. Computers & Security, Volume 9, 2 May 2020.
28. Rahim Taheri, Meysam Ghahramani, Reza Javidan, Mohammad Shojafar, Mauro Conti. Similarity-based Android malware detection using Hamming distance of static binary features. Future Generation Computer Systems, Volume 10, 5 April 2020, Pages 230-247.
29. ElMouatez Billah Karbab, Mourad Debbabi, Abdelouahid Derhab, and Djedjiga Mouheb. MalDozer: Automatic framework for android malware detection using deep learning. Digital Investigation, Volume 24, Supplement, March 2018, Pages s48-s59.