Machine Learning Techniques for Malware Detection with Challenges and Future Directions

Mohammed A. Alqahtani[1]

[1]Department of Computer Information Systems, College of Computer Science and Information Technology,
Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia

***Abstract*:** In the recent times Cybersecurity is the hot research topic because of its sensitivity. Especially at the times of digital world where everything is now transformed into digital medium. All the critical transactions are being carried out online with internet applications. Malware is an important issue which has the capability of stealing the privacy and funds from an ordinary person who is doing sensitive transactions through his mobile device. Researchers in the current time are striving to develop efficient techniques to detect these kinds of attacks. Not only individuals are getting offended even the governments are getting effected by these kinds of attacks and losing big amount of funds. In this work various Artificial intelligent and machine learning techniques are discussed which were implements for the detection of malware. Traditional machine learning techniques like Decision tree, K-Nearest Neighbor and Support vector machine and further to advanced machine learning techniques like Artificial neural network and convolution neural network are discussed. Among the discussed techniques, the work got the highest accuracy is 99% followed by 98.422%, 97.3% and 96% where the authors have implemented package-level API calls as feature, followed by advanced classification technique. Also, dataset details are discussed and listed which were used for the experimentation of malware detection, among the many dataset DREBIN had the most significant number of samples with 123453 Benign samples and 5560 Malware samples. Finally, open challenges are listed, and the future directions are highlighted which would encourage a new researcher to adopt this field of research and solve these open challenges with the help of future direction details provided in this paper. The paper is concluded with the limitation and conclusion section.

***Keywords*:** Artificial neural network, convolution neural network, cybersecurity, malware, machine learning, support vector machine.

1. Introduction

Technologies related to smart cities are combined with Android OS applications as these applications are supporting the smart city requirements. Android is playing an essential role in various fields in the current development phase of the earth. These applications are helping different sectors like competent government, intelligent transportation, and other energy resources management [1][2]. The facilities provided by Android are allowing developers to utilize them in their applications. Still, at the same time, it is acting as a source to attackers like malware which is targeting the Android to cause serious threats which are leading to financial loss, leakage of critical data, and security concerns for nations [3]. Based on the excellent features offered by Android and flexible to adopt technologies, it has captured as much as 80% of smartphone users. But, at the same time, it has become a significant source for malware attacks [4]. Based on this report [5], as many as four million fresh malicious applications were developed during 2019. The average time in which the hackers create an infected APP is around every eight seconds. This statistic of malware attacks and the development of malicious applications are alarming and becoming a source of a significant threat to cyber-security. So, well-managed and efficient detection methods are need of the hour for Android malware, and it needs urgent attention from computer professionals to develop tools to detect these attacks significantly to improve mobile security [6]. Another study conducted by Juniper Networks Mobile Threat Centre (MTC) [7] has reported as many as 155 % of the increase in mobile malware attacks in 2011 compared to previous years in almost all the platforms. Also, with regards to Android malware, there is an increase of about 3000 % during 2011. Based on these statistics and numbers, a significant increase in malware was reported in third-party Android applications enjoying the same privileges as an actual application available in the Google Play Store. The reason for this increase in the numbers is, in the past, any Android developer had the privilege to develop an application and post them immediately at the official Android Market without screening the application for the presence of malware and inspecting the application. Another reason for the drastic increase in malware is due to the blending of Google Android dominant market and smartphones and their share, which was around 68.8% during 2012, and the absence of efficient security control methodology to control the applications which are being posted on Android application markets. In a recent study, the reports state that as many as 700,000 apps have crossed 15 billion downloads in the Google Play Store. As these numbers are increasing so also the money stealing malware is also growing based on the security firm Fortinet which was done during 2006-2011.

Malware detection methods are classified into two categories based on the process by which they are detected, like signature-based method or behavior-based method. In the current scenario, signature-based malware detection is working fine for the previously detected malware, which was done with the help of anti-malware vendors. But the polymorphic malware is not yet detected, which can change the signatures. Also, new malware which is expected to occur in the future cannot be detected by these traditional methodologies. So, the best solution that is recommended by researchers is utilizing the heuristic analysis along with the machine learning techniques that could provide better solutions with higher efficiency for detection. As practice has shown, the traditional approach to the field of malware detection, which is based on signature analysis [8], is not acceptable for detecting unknown computer viruses.

In the current times, mobile security researchers are working hard to counter the risk of malicious Apps. They are proposing various defense approaches especially novel methodologies to detect malware. These methodologies are based on machine learning techniques which are proving to be very efficient in various research fields in solving their problems. Further to machine learning, they are also implementing advanced techniques like deep learning, which is attracting security researchers of multiple areas to implement this technique [9] [10]. The process of utilizing machine learning techniques for the detection of state-of-the-art malware attacks could be generally classified into three categories: Static feature [11] [12], dynamic feature [13] [14], and mixed feature [15] [16]. The most critical aspect of machine learning to get good accuracy in the result is feature selection. It is also used during training the model. The more accurate the features are, the more robust the model is, and the more would be the classification accuracy [17] [18] [19]. The feature selection is a crucial step in machine learning to enhance the performance of the model and also to make a robust model all the irrelevant features and redundant features should be removed, and sophisticated features which play a significant role should be selected. This process is time-consuming as the result of machine learning depends on this step. The outline of this paper is as shown in Fig. 1.
The significant contributions of this study are as follows:

- Databases related to malware detection are listed in the paper along with the methodologies and techniques applied on these datasets to detect malware.
- The author has collected the relevant techniques for the detection of malware.
- The techniques focused in this study are tradition classification, Artificial neural network and Convolution neural network
- Relevant and latest features extraction techniques applied for the classification of malware are discussed with their results.
- The author has listed the current open challenges for detecting malware.
- The author has listed future directions for a potential researcher to motivate and provide enough information to take this research work further by solving the open challenges.
- The author tried to represent all the essential information into tabular format with the reference number.

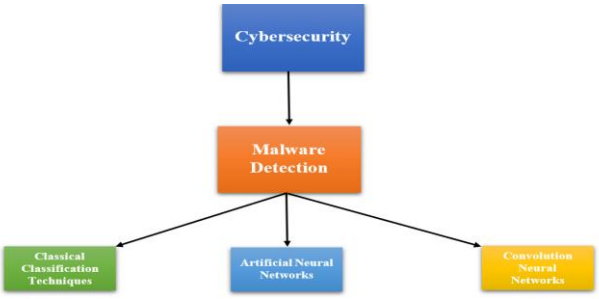

Figure 1. General overview of the paper

The rest of the paper is as follows: section 1 gives the complete introduction about the research field, section 2 discusses the primary classification techniques with their technical details, section 3 provides the database with more information which is applied for malware detection, and the researchers who used these databases for experimentation, section 4 gives the detail description of researchers who applied various classification technique to detect malware, section 5 discusses the open challenges and future scope of the work, section 6 has the conclusion section, followed by section 7 as limitations and 8 with future scope, followed by the reference list.

2. Related Work

Machine learning techniques are currently applied in various applications. So, Malware detection is no exception. It is playing an essential role in detecting malware more efficiently compared to traditional techniques. In this section, malware detection based on various machine learning techniques like Support vector machine, Artificial neural network, and Convolution neural network is discussed with the features extraction details and the results achieved.

2.1 Malware detection based on classical techniques

The authors in this work [20] introduced a methodology that has the capability of extracting the features automatically. The system was developed using JavaScript, and the detection is static. This system takes the input as .apk Android executable files. The following are the features extracted by the system: API count for each method related to phone management and API code of relation with phone control and API related to privacy information. Further, after the extraction of features, the author has used the J48 Decision Tree classification method for classifying using the Weka library. The authors followed 10-fold cross-validation. The ultimate results of this methodology are 82.7 % accuracy in classifying the malware, and the false-negative rate is 17.3%.
In another work [21], a rooted Android device was used with Linux-based tools to capture the calls received on the system. The feature vectors were collected by capturing the number from each call received by the system with the help of the Android application. The following were the most relevant system calls captured based on the experimentation: read (), open(), access(), and chow(). To distinguish between a benign and malware version, the authors applied a 2-means clustering algorithm.
The authors in this work [22] designed a real-time anomaly detector. The technique is based on a 1-Nearest Neighbor classifier. The total number of features extracted was 13. Two features were examined at the user's end depending on the phone's active or inactive position and based on the SMS which are sent while the phone was inactive. The methodology was based on two models, first to capture the features at every 1-second interval, and the other is captured at every one-minute interval. The system was evaluated for ten genuine malware samples, and the authors claim to detect as much as 93% of malware detected correctly, while the false-positive rate was 0.0001.
In another work [23], the authors worked on 88 features with an unrooted device. These features were experimented on two machines which were real and among two different users. The testing was done with 16 benign and 4 self-made malware. The features were monitored by the model at an interval for every 2 seconds. The following are the classifiers used by the authors k-Means, Logistic Regression, Histograms, Decision Tree, Bayesian Networks, and Naive Bayes. Apart from the classifiers, the authors also applied

filters for the selection of relevant features by comparing them with Chi-Square, Fisher Score, and Information Gain. Based on the experimentation, the authors claim 80% detection of malware with the available 88 features. The false-positive rate was 0.12.

The authors in this work [24] also did a comparative analysis between different classification algorithms. They used 500 Android .apk files to make the classification and processing. In this work, 160 permissions were used as feature vectors. The comparison was made among Random Forest, J48 Decision Tree, and Classification and Regression Tree (CART) algorithms. But unfortunately, the results were not claimed.

A malware detection study was conducted by [26] for Android platform. The classification technique applied here is SVM, and the features used for classification were a combination of vulnerable API calls and risky permission of calls. Extensive experimentations were carried to validate the proposed methodology which can detect the malware and further identify the malicious Android applications more efficiently. The dataset used for this experiment was taken from [27]. The experiments were also conducted on datasets downloaded from benign apps from Google play to validate and do some comparative analysis.

Similarly, in this study [25] data mining technique is applied to detect the malicious software and the authors applied experimentation and investigation to detect the malware by applying the SVM algorithm. The purpose of this work was to detect the malware rate for the SVM method. The result of this study after applying SVM algorithm was the probability of detecting around 74-83% malware. The detection is done by the models developed with SVM and using enough datasets of malicious software's. Related Techniques are listed in table 1.

2.2 Malware detection based on artificial neural networks

The authors in this work [28], investigate the behavior of malicious data. The experimentation was conducted on features extracted from Global wavelet transform and GIST. The dataset used in this experimentation was Mahenur, it consists of 3131 samples of binaries with 24 unique malware families. Feedforward Artificial Neural Network technique was applied for the experiment. The authors claim up to 96.35% of accuracy in detecting and classifying the malware.

In this work [29], the authors have applied a sequence mining algorithm for the detection of malware patterns. The feature used in this experimentation is the instruction sequence extracted from the sample files. Then, All-Nearest-Neighbor (ANN) classifier is applied to detect the malware based on the features extracted in the patterns form. The framework was based on pattern mining methodology along with ANN classifier. The proposed method is then evaluated by the real and unseen malware samples. The results were outstanding; wherein this method outperformed all the other alternate data mining approaches. The dataset used to do the experimentation was collected from Windows PE samples which were 10,307 among which 8847 were malicious files, and 1460 were benign files.

The authors in this work [30] have applied various classical machine learning classifiers also deep neural network and proposed a system to detect malware attacks on Android systems. The significant contribution was extracting the relevant features, which are also sensitive. They have designed and developed a static analysis tool for the feature extraction step. The outcome of this approach is around 97.3% of accuracy for the true positive rate.

In another work [31] for malware detection, the authors here have applied a different approach wherein the sensitive packages which cause the malware attacks are detected. These are called package-level API calls, which are found in common inside malicious apps. This approach is also capable of extracting another package level that gives the feature details for a large set of Android malware. This technique gave good results with around 99% of accuracy and a false positive rate of 2.2%.

In another work [27], the significant contribution was working on extracting features. The features were extracted from the manifest file and from the source code of the application. These features were then trained with an SVM classifier. The classifier was able to classify and detect malware based on the features provided, and further, it achieved good accuracy and performance for the detection of malware.

The authors in this work applied a different approach to detect the malware [32], semantic features were extracted from the weighted contextual API. These APIs were used to draw dependency graphs further to classify them as Android malware or not. The proposed system is capable of effectively fighting against malware.

Another feature extraction technique was proposed by [33], to detect malware. Here the authors have applied the Markov chain to the sequence of API calls and then extract the relevant features. After removing these features, then traditional classification methodology is used to classify and detect the malware.

In this work [34], features were extracted from permissions and API calls. These features were extracted from the files stored in the Android manifest file. These extracted features were then applied for classification using the K-nearest neighbor algorithm. These classifiers are used to classify benign and malware apps. The results were further improved after applying the K-means algorithm. The results are summaries as shown in Tab. 2.

2.3 Malware detection based on convolution neural network

In recent times CNN is widely used in many applications because of its capability to handle big datasets and pattern-matching tasks. The latest application of CNN is on IoT and the usage of the internet by the common users. These internet facilities have attracted many users and collected Hugh amount of data and at the same time opened gates for numerous vulnerabilities for attackers and make cyber-attacks. There is no robust security solution to handle these kinds of attacks and detection methodologies for the developing IoT environments, which has invited many developers and researchers to find efficient solutions for numerous DDoS attacks. So, CNN plays a vital role in this aspect by detecting the possible DDoS malware attacks for large datasets using CNN-based algorithms. The following are few works done by the researcher for detecting malware using CNN algorithms.

In this work [35], the authors have proposed a methodology to detect malware. The approach followed is a light-weight malware detection, wherein initially, the malware is converted into a one-channel gray-scale image. These

images are then used to extract features that ultimately represent light-weight CNN to detect the malware and its family. The outcome of this methodology is the accuracy of 94.0% under the two-class classification, while the accuracy achieved was 81.8% for the three-class classification. These results were then compared with [36] with a similar methodology for the classification of malware. The difference among these two methodologies was the difference in the number of
layers as [36] used very deep network, and the pre-processing step was very complex.

Another similar work is conducted by [35], wherein the authors took raw features for classification. This method can detect only two types of malware very effectively because of the simplicity of the classification method.

This methodology is very suitable for IoT devices wherein only the first layer is used for malware detection. Some other works also represent similar kinds of contributions where a great deal of research is done based on neural network [37] [38] [39]. Similar work was done by [40], by utilizing CNN with a large-scale methodology. This technique was developed to classify malware based on random projections. However, the accuracy did not improve even after increasing the number of hidden layers. In another work [41], the authors applied feed-forward DNNs for malware detection and classification using static methodology. But, they could not provide significant results and improvement inaccuracy. Multi-tasking approach was introduced by [42], wherein the authors applied multi-tasking learning with the combination of feedforward and neural network. The number of layers utilized in this approach was four hidden layers, but the results were not discussed very clearly.

In recent types, the authors in this work [43] have introduced a feature extraction technique (word2vec). These features were extracted based on DL and represent more like malware which is based on opcodes. The proposed methodology included a gradient boosting algorithm and k-fold cross-validation to assess and evaluate the methodology's performance. The dataset used in this experimentation was minimal, and the authors claim a 96% accuracy which seems to be better than other similar works.

In this work [44], the authors introduced a new approach of converting the API call sequence into RGB images. This conversion is done by mapping the corresponding protection levels depending on the permissions required for the API call. Then these API call sequences, along with the protection level, were converted into RGB images. The authors have used CNN and developed a model to detect the malware. The accuracy produced by this methodology was around 93%. The dataset used for experimentation was 7192 benign and 24461 malicious samples.

In another work [45], a similar strategy was applied for feature extraction. Here the authors have utilized the static method to extract the features. The authors have extracted 138 features into four different categories and then converted these permissions features into 12x12 PNG images. Then, CNN is applied to train the model and detect the malware. For the experimentation 2500, Android applications were used, and the dataset had 2000 malicious and 500 benign samples. The results achieved were 93% accuracy in detecting the malware.

Similarly, in this work [46], the authors also converted the .dex files into RGB color codes and then into color images. These images were at a fixed size. Then the images were put into CNN as inputs for the extraction of automatic features and further to the training process. Here the dataset used for experimentation was 2 million benign and malicious Android applications. The outcome if this experimentation was accuracy of 98.4225% of malware detection.

Another methodology to extract features was introduced in [47], where texture fingerprints were applied to remove the malware features. The texture image information extracted from the sample applications codes was utilized in this methodology. To detect the Android malware, these codes were mapped with the uncompressed gray value, and further, they were combined with the API. These combinations of code and API call features led to the detection of malware. The technique applied in this methodology is a deep belief network. The dataset used contained 6956 samples and the proposed model yield 95.6% of malware detection accuracy.

The authors in this work [48], did similar work by converting the APK code into images and then applying CNN to classify the images and detect the malware. The dataset used in this experimentation was 720 benign samples and 720 malicious samples. The model got an accuracy of 92.67% for detecting the malware. In another work [49], the semantic graph technique is applied to extract the features from the Android application, and then CNN is used to train the model. The dataset is collected from various sources like Marvin, Drebin, VirusShare, and ContagioDump. This methodology got an accuracy of 99% in detecting the malware. In another graph-based technique [50], the features were extracted from API call graphs and applied the features to deep neural network, and the dataset consisted of 33,139 malicious and 25,000 benign samples. The results got 98.9% of accuracy in detecting the malware. The summary of the above-discussed techniques is as shown in Tab. 3.

In this work [51], a dynamic android malware analysis technique is proposed for the detection of possible malware. Here the authors have constructed OS-level and Java-level semantic views. The system is capable of tracking the changes in the files such as threads and processes. It can detect the malware through system calls and Dalvik instructions. These details are provided for the dynamic analysis.

3. Datasets

To effectively evaluate the performance of the classical machine learning and deep learning techniques, a large dataset is needed with relevant information consisting of various types of samples. Unfortunately, there is no such dataset available publicly to conduct research in cybersecurity and malware detection. The reason for this unavailability is the privacy-preserving policies. Since each organization has some individual privacy policies, it is difficult to collect a large dataset for malware detection. In the current times, malware has increased, as statistics show, keeping all these growing malware and their families together in one place is a difficult task. Although researchers are making a significant contribution to this field but still their findings are not recognized because of the shortage of a single dataset which can accommodate all the required samples.

So, there is a need for such repositories to help the researchers to accomplish their tasks more efficiently. There are many researchers who could manage to acquire some data for processing and detecting the malware, so here are

some of the works that have used some available datasets and their training and testing details.

In this work [52], the authors have utilized the publicly available dataset named Ember. This dataset consists of both malicious and benign files. It has 70,140 benign files and 69,860 malicious files. These files are used to train the model, and the dataset is divided into training and testing files. They have divided the dataset as 60% for training and 40% as testing by applying Scikit-learn. The number of files that came under training is 42,140 benign while 41,860 as malicious files, while for the testing dataset 28,000 files for benign and 28,000 as malicious.

In another work [53], the authors tried to find the kind of malware or to determine the family with which this malware belongs. To evaluate the findings, they have utilized the dataset in two formats, one set which consists of both the benign and malicious files together and in another set only malicious files the details are as shown in Tab. 4 and Tab. 5, respectively. The authors tried to find whether a given app is malicious or not? And look for the most relevant attribute which can help in determining the family of the detected malware. The experiments were conducted, and the results were evaluated on two datasets Malgenome [54] and Drebin [27]. These two datasets made around 33,000 files combined from the two datasets. The authors also tried to validate their findings with another two more datasets acquired from virus share.com, Contagio Minidump [55].

The authors in this work [56] did a similar study wherein they developed a model to detect whether the downloaded application is benign or malware. They have used an online malware scanning application called VirusTotal [57] in the experimentation stages. To evaluate their work 20,000 malware samples were used from VirusShare [58] and 1,260 from the Malgenome project files [59], and additionally 20,000 files from Google Play App Store [60]; these files were selected during the time period of March 2015 till April 2016.

It is a fact that most of Android application are non-malicious. This reality is becoming a challenge for Android malware detection, especially for deep learning technique. To detect Android malware, large amount of data is needed, and these data samples should be constantly updated with malicious samples. This would help the deep learning models to detect the Android malware. But unfortunately, most of the datasets are not much large, which is a challenge for deep learning experts. The following are some of the datasets available for researchers to use with their technical details along with their acquisition dates.

DREBIN dataset [27], is considered as the commonly used dataset for detecting Android malware, it consists of 5,560 malicious samples while 123,453 benign samples. This dataset was collected during August 2010 till October 2012.

Another dataset available for malware detection is Android Malware Genome Project data [63].

This dataset consists of 1260 samples related to malicious while 863 benign samples. The malicious samples are grouped into 49 different categories. The dates of its acquisition are from August 2010 until October 2011.

Contagio [64], is another available dataset for malware detection. It has around 1150 malicious samples. The collection time for this dataset is 2011.

In its additional version [65] this dataset consisted of about 24553 malicious samples. These samples were grouped into 135 categories related to the malicious family. The collection period of these samples where from 2010 till 2016.

Another dataset related to malware detection is available, called as Android PRAGuard dataset [66].

The total malicious sample is 10479, which were collected in the year 2015. Similarly, the Marvin dataset [16] is also available for malware detection. It is a mixed dataset consisting of both malicious and benign files together. It has 10572 malicious samples and 75996 benign samples. The collection time is in the year 2015. Another available dataset is ISCX Android Botnet Dataset [67]. It consists of 1929 malicious samples. These samples are categorized into 14 different categories. The collection time is from 2010 till 2014. The summary of the above discussed datasets is as shown in Tab. 6.

4. Methodologies

The technical details related to different machine learning techniques with the equations and the details related to the equation for SVM, ANN, and CNN.

4.1 Decision tree

A Decision Tree is a tree-based classification in structure wherein each vertex of the tree is considered as an attribute, and the corresponding branch gives the value of that attribute [53]. The root is the topmost vertex of the tree, which stores the most important information. The difference in the entropy is stored on the root of the tree. This root stores the critical feature which is used for splitting the training data in the most optimal way. The bottom nodes of the tree are known as leaves, as shown in Fig. 2. The classes are represented as the leaves at the bottom of the tree. The process of classification is traversing the decision tree from top to bottom of the tree by satisfying the instance needed to classify. The equation of information gain used in a DT to optimally split instances in a tree-structured manner is given below.

$$Gain(P,Q) = Entropy(P) - \sum_{v \in D_Q} \frac{|P_v|}{|P|} Entropy(P_v) \quad (1)$$

Here, Gain(P,Q) is the reduction in entropy in order to sort P on attribute Q as in equation 1. Features with increasing information gain value are chosen as nodes in a top-down manner.

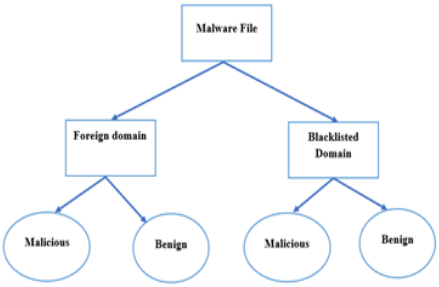

Figure 2. Decision tree architecture

4.2 Support vector machine

In cybersecurity, SVM is considered as the most popular classification algorithm. It is a supervised learning algorithm. The ultimate goal of applying this algorithm is to separate the hyperplane in the feature space among different classes [69]. The hyperplane is the main constraint in getting optimized results. In this technique, the hyperplane is chosen based on the distance between the hyperplanes, which should be maximum for the closest data points. As shown in

the Fig. 3. The following are the parameters b, w, x as in equation 2. Where b is the hyperplane and w is the weight and x is the data points. As shown in the equation points (x1, y1), (x2, y2),: : : ,(xn, yn). Here x is an element of real values R and y = (1, 1) aslabels. The ultimate target of SVM is to precisely classify the training data where y = + 1 using wxi + b ≥ 1 and when y = - 1 using wxi + b ≤ 1. So, for all i, yi(wxi + b) ≥ 1 using the distance measure performed by the following:

$$M = \frac{2}{|w|} \qquad (2)$$

The most important advantage of SVM is it ability to classify the data more accurately and the simplicity in implementation when compared with other classification algorithms. The accuracy is very good when the number of features (m) are higher than the number of samples (n) in the dataset. These numbers could be represented as m >>n. SVM is very much utilized in cybersecurity and also in other fields of research like healthcare, biology, and pattern recognition.

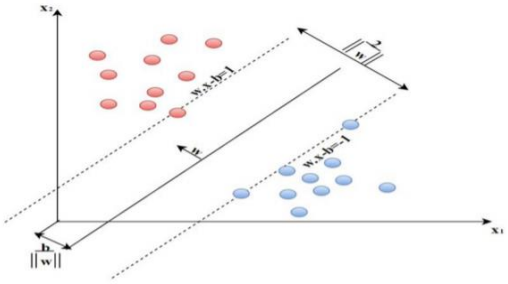

Figure 3. SVM architecture

Further to the advantages of SVM it can create better hyperplanes with the time complexity of O(N2)[70]. The comparison between all these techniques are listed in Tab. 7.

4.3 Artificial neural network

Artificial neural network (ANNs) are a collection of nodes. The neurons of the brain simulate these nodes. ANN is a combination of three basic layers, which are named input, hidden and output layers. The hidden layer can be changed and increased to more than one layer, depending on the design of the algorithm. The sequence of operation is as follows: initially, the input layer transfers its output to the hidden layer, and accordingly, each subsequent layer passes it output to the next layer, and ultimately the final output is passed to the output layer, and this is the results of the classification as shown in Fig. 4. Before SVM was invented during the 1990s ANN was very popular at that time, but further to the enhancement of ANN with feed-forward and convolutional neural network, again, ANN gained its popularity. ANN is very much utilized in the cybersecurity field. The learning process is the main phase in any machine learning algorithm so in ANN also, this process takes inputs (x1, x2,: : :, xn) with a given output label as y. Then weight vectors (w1,w2, : :,wn) are used to weight the input vectors. The weights play an important role in the learning process, as these are adjusted in such a way that the learning error is minimized E = ∑_(i=1)^n⁡〖ld_i-y_i l〗 , where the error is defined as the difference between the actual output (yi) of the neuron and the output which is desired (di). The adjustment of the weights is made by gradient algorithm, which follows back-propagation methodology, wherein the learning process is repeated backward and forward directions and the error is followed, this process is done until the error shows lower than the stipulated threshold value. The adjustment of the weights is made based on the following equation

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j} \qquad (3)$$

Δw_(i,j)= ηδjxi.j, i is the input node, and j is the hidden node. The parameters are shown in equation 3.

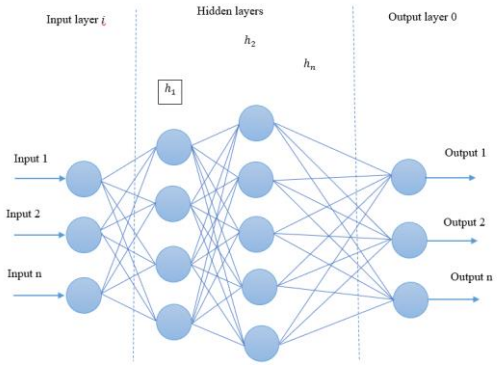

Figure 4. Artificial Neural Network architecture

4.4 Convolution neural network

Deep Learning is a subclass of Machine Learning, it is basically applied to handle applications with large training datasets. The process followed by DL is hierarchical in nature which is based on feature abstraction and feature representation. In the contrary, traditional ML algorithms performance gets degraded when the training dataset is very large and because of the dimensionality of the data. Hence, to resolve this issue of data size, DL algorithms are applied, which usually use graphical processing units (GPUs) for processing the big data. Convolution neural network is the most applied algorithm among DL algorithms, especially to handle cybersecurity applications. CNN is composed of two main layers: Convolution layer and the Pooling layer. The main functionality of convolution layer is to convolute the input data. This convolution is done with the assistance of numerous similar-size kernels. The outcome of this convolution is the retrieval of features from the input data. These features are the outcome of applying high value to a given position if the desired feature is available at that position and location and also in the reverse form [71]. The following are the main parameters in calculating the optimized features: m for kernel width and height, h is the output for the convolution, x represents the input, and finally, w is the convolution kernel as shown in the eq 4.

$$h_{i,j} = \sum_{k=1}^{m}\sum_{l=1}^{m} w_{k,l}x_i + k - 1, j + l - 1 \qquad (4)$$

The next layer in CNN is the pooling layer, it is used to down-sample the feature sizes. It is done by applying two types of pooling techniques one if max-pooling and the other is average pooling. Max-pooling selects the value which is maximum among the values retrieved from the previous layer. The average pooling takes the average values of the last layer values. The pooling mechanism is applied to get the maximum and average deals from the previous layers which is under the kernel value [71]. Mathematically as in equation 5:

$$h_{i,j} = \max\{x_{i+k-1,j+l-1}\ \forall 1\ k\ m, 1\ l\ m\} \qquad (5)$$

Apart from the above-mentioned layers, CNN has activation functions also. The most commonly used activation function is rectified linear unit. This activation function is represented

by the equation f (x) = max(0, a). A typical CNN architecture is as shown in the below Fig. 5. CNN has many advantages over other algorithms except computational cost, which is a disadvantage of CNN

5. Open challenges and future research directions

Cybersecurity is a hot research topic although lot of work has been done in the past. But, still there are many open challenges available and need to be addressed. In this section few of them are highlighted and future directions are listed to help a potential researcher to focus on these issues and adopt these directions. In Tab. 8 all the open challenges and future directions are listed.

5.1 Performance evaluation framework

The evaluation metric used to calculate the performance of the cybersecurity technique and accuracy of the proposed methodology needs standardizations. There needs robust evaluation metrics. Since the recently proposed evaluation metric does not cover all the aspects of cybersecurity especially with different domains. This is making it difficult to compare the proposed methodology with system of different domain. The challenge is to develop the best way to evaluate the proposed method and compare it with state-of-the-art techniques. A universal evaluation method is needed to check the robustness and performance of the proposed technique under different scenarios. The research community needs such evaluation techniques that could confirm the robustness and performance capability of the current and future methodologies among different scenarios and domains. The following are the three research directions were researchers can explore further to overcome this challenge: Developing tools and protocols and checking them with the attack and without attacks and see the performance, Listing the common criteria for efficient performance and document them, establishing open online platform for other researchers to suggest their methodologies and let them participate in this cause.

5.2 Generalization

Generalization concerning the proposed techniques is another challenge. In this, the proposed methodology doesn't perform well under different circumstances like failing with novel samples, attacks, and dataset contents. These models are trained on specific training data, which is tuned, and the model is proposed, which fails to counter any change in the testing data if any real issues come like novel malware samples. The performance of the model degrades at some points, which is a big challenge and needs a solution. Additionally, the proposed cybersecurity model for ransomware attacks would fail if tested for spyware-related attacks. Cybersecurity models proposed are usually trained on known threats, which if tested on actual samples, would fail to perform. Since the nature of the attack is not predicted, it is now a challenge for researchers to counter this issue. The following are the research direction which would help the researcher in exploring their research in this direction: Exploring a wide range of attacks and developing models based on this study and proposing mathematical security models which are capable to handle these attacks efficiently.

5.3 Design of Security

The traditional systems proposed for cybersecurity systems which start with data collections till the classification followed by feature extraction, should be rechecked with the consideration of adversaries. For example, the features extracted should be not only generalized but also capable of handling the vulnerabilities to attacks. The solution to this challenge would be updating the system occasionally by adding the latest features relevant to the attack. These features should be able to solve the computational complexity issues and be faster and automated.

5.4 Advanced machine learning

Traditional machine learning techniques applied for cybersecurity issues with systems built on modeling non-linear adversary behavior and not trustworthy features lead to over-fitting and degrading performance and reliability. This is a challenge and needs attention from researchers. To overcome this crisis, Advanced Machine Learning (AML) needs to be developed. Currently, a minimal amount of work has been done on AML like recognition of features, dictionary learning and DL for cybersecurity. The future scope of AML is to explore robust feature extraction techniques for different datasets.

5.5 Robustness of security with DL

Although DNN systems give high accuracy for predicting malware attacks and other security tasks, recent studies have shown these techniques to be vulnerable for inputs with subtle perturbation. These adversarial examples are acting as real threats to DL techniques and need a proper solution. The scientific reason for this issue is studied by few researchers who say that the linear nature of the DNN-classifier is the primary source of the problem. The solution is to develop more robust DNN models, and there is a vast scope in creating these kinds of models which can handle adversarial issues more efficiently.

5.6 Privacy preserving in cybersecurity

In recent research, it is evident that ML and DL-based cybersecurity systems have proved to be very efficient in classifying malware. But, on the other hand, these techniques could not handle the privacy issues related to datasets as the privacy of the datasets used in the systems is getting leaked and no proper methodology is available to address this issue. If any advanced techniques are applied to save the leakage of private data, then the accuracy is getting affected. Therefore, more research is needing to keep the privacy-preserving issue and not affecting the accuracy of the proposed method.

5.7 Encyclopedic datasets

Cybersecurity is heavily dependent on real datasets. There are few existing datasets for cybersecurity, but they lack in big sizes and rich features like fully labelled, structured and complete with diversity and details about the attacks, their domains, and usage capabilities that should resemble the actual data. The following are the biggest hurdling in creating such datasets: high reliability, difficulty creating accurate and true labels, and access to applications in natural environments. These things needed to be taken care to develop a more robust dataset to handle real cybersecurity issues.

5.8 Interdisciplinary research

Cybersecurity technologies should be merged with different technologies and make it a multidisciplinary field of research: there and many advantages in doing so. Cybersecurity research along with fundamental research, including a significant contribution from computer science, ML, and psychology, should be done together. These multidisciplinary modes of research would enhance the quality by increasing the reliability and development of efficient methods against numerous security threats and attacks."

6. Conclusions and Future work

Cybersecurity is a real research topic wherein real problems are encounters that reflect directly to common people and some government authorities. Malware is threats for mobile users wherein critical and sensitive information is shared and downloaded. So, the attackers are targeting mobile devices in the form of malware attacks. Researchers in the current time are developing the latest techniques to detect this malware. In this context, this paper gives and detailed information about the latest techniques and their outcome. Machine learning techniques are now implemented almost all the fields as it is showing promising results. So, in this paper, machine learning techniques that are applied to detect malware are discussed. The datasets on which the experimentation is conducted are discussed and listed to help future researchers. The traditional machine learning techniques like Decision tree, Support vector machine are discussed with the feature extraction techniques and their details, followed by the latest convolution neural network techniques. Finally, a list of open challenges is discussed, and future directions are suggested to help a potential researcher to follow.

In this work, various machine learning techniques are discussed, which were implemented for detecting malware attacks. But, some of the methods which are discussed here did not disclose the results, and some of the techniques do not discuss the features, and some of them do not show the actual dataset used for the experimentation.

As some of the discussed technique lack few important information. In the future, this information could be explored further to draft a complete report about the available datasets and their details of acquisitions, and complete information about feature extraction techniques in a systematic format and then categized based on the similarity of extraction and feature details. Then machine learning techniques could be explored more for recent updates and possible extra information for the readers to get the most recent information.

7. Acknowledgement

The author would like to thank SAUDI ARAMCO Cybersecurity Chair for funding this project.

References

[1] M. Daraghmeh, I. Al Ridhawi, M. Aloqaily, Y. Jararweh and A. Agarwal, "A power management approach to reduce energy consumption for edge computing servers," in Proc. FMEC, Rome, Italy, pp. 259–264, 2019.

[2] F. A. Turjman, H. Zahmatkesh and L. Mostarda, "Quantifying uncertainty in internet of medical things and big-data services using intelligence and deep learning," IEEE Access, vol. 7, pp. 115749–115759, 2019.

[3] M. N. Alenezi, H. Alabdulrazzaq, A. A. Alshaher, and M. M. Alkharang. "Evolution of Malware Threats and Techniques: A Review," International Journal of Communication Networks and Information Security, vol. 12, no. 3, pp. 326-337, 2020.

[4] M. Nassiri, H. HaddadPajouh, A. Dehghantanha, H. Karimipour, R. M. Parizi and G. Srivastava, "Malware elimination impact on dynamic analysis: An experimental machine learning approach," in Handbook of Big Data Privacy, Springer, pp. 359–370, 2020.

[5] B. D. Deebak, F. A. Turjman, M. Aloqaily and O. Alfandi, "An authentic-based privacy preservation protocol for smart e-healthcare systems in IoT," IEEE Access, vol. 7, pp. 135632–135649, 2019.

[6] Z. Lv, W. Mazurczyk, S. Wendzel and H. Song, "Guest Editorial: Recent Advances in Cyber-Physical Security in Industrial Environments," IEEE Transactions on Industrial Informatics, vol. 15, no. 12, pp. 6468–6471, 2019.

[7] "Juniper, 'Juniper Networks 2011 Mobile Threats Report,' Juniper Networks Mobile Threat Center (MTC), 2012."

[8] I. Martin, J. A. Hernández and S. de los Santos, "Machine-Learning based analysis and classification of Android malware signatures," Future Generation Computer Systems, vol. 97, no. 1, pp. 295–305, 2019.

[9] K. Xu, Y. Li, R. H. Deng and K. Chen, "Deeprefiner: Multi-layer android malware detection system applying deep neural networks," in Proc. EuroS&P, London, U.K, pp. 473–487, 2018.

[10] M. Amin, T. A. Tanveer, M. Tehseen, M. Khan, F. A. Khan et al., "Static malware detection and attribution in android byte-code through an end-to-end deep system," Future Generation Computer Systems, vol. 102, no. 3, pp. 112–126, 2020.

[11] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han and X. Zhang, "Exploring permission-induced risk in android applications for malicious application detection," IEEE Transactions on Information Forensics and Security, vol. 9, no. 11, pp. 1869–1882, 2014.

[12] P. Faruki, V. Ganmoor, V. Laxmi, M. S. Gaur and A. Bharmal, "AndroSimilar: robust statistical feature signature for Android malware detection," in Proc. SIN, Aksaray Turkey, pp. 152–159, 2013.

[13] M. Y. Wong and D. Lie, "IntelliDroid: A Targeted Input Generator for the Dynamic Analysis of Android Malware.," in Proc. NDSS, San Diego, California, USA, pp. 21–24, 2016.

[14] H. Cai, N. Meng, B. Ryder and D. Yao, "Droidcat: Effective android malware detection and categorization via app-level profiling," IEEE Transactions on Information Forensics and Security, vol. 14, no. 6, pp. 1455–1470, 2018.

[15] S. Chen, M. Xue, Z. Tang, L. Xu and H. Zhu, "Stormdroid: A streaminglized machine learning-based system for detecting android malware," in Proc. ASIS CCS, Xi'an China, pp. 377–388, 2016.

[16] M. Lindorfer, M. Neugschwandtner and C. Platzer, "Marvin: Efficient and comprehensive mobile app classification through static and dynamic analysis," in Proc. COMPSAC, Taichung, Taiwan, pp. 422–433, 2015.

[17] P. Vinod, A. Zemmari and M. Conti, "A machine learning based approach to detect malicious android apps using discriminant system calls," Future Generation Computer Systems, vol. 94, no. 5, pp. 333–350, 2019.

[18] D. Ucci, L. Aniello and R. Baldoni, "Survey of machine learning techniques for malware analysis," Computers & Security, vol. 81, no. 6, pp. 123–147, 2019.

[19] P. Feng, J. Ma, C. Sun, X. Xu and Y. Ma, "A novel dynamic Android malware detection system with ensemble learning," IEEE Access, vol. 6, pp. 30996–31011, 2018.

[20] D. Kim, J. Kim and S. Kim, "A malicious application detection framework using automatic feature extraction tool on android market," In Proc. ICCSIE, Xian, China, 2013.

[21] I. Burguera, U. Zurutuza and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for android," in Proc. SPAM, Chicago Illinois USA, pp. 15–26, 2011.

[22] G. Dini, F. Martinelli, A. Saracino and D. Sgandurra, "MADAM: a multi-level anomaly detector for android malware," in Proc. MMM-ACNS, St. Petersburg, Russia, pp. 240–253, 2012.

[23] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer and Y. Weiss, "'Andromaly': a behavioral malware detection framework for android devices," Journal of Intelligent Information Systems, vol. 38, no. 1, pp. 161–190, 2012.

[24] W. Z. Zarni Aung, "Permission-based android malware detection," International Journal of Scientific & Technology Research, vol. 2, no. 3, pp. 228–234, 2013.

[25] B. Sanjaa and E. Chuluun, "Malware detection using linear SVM," in Ifost, vol. 2, pp. 136–138, 2013.

[26] W. Li, J. Ge and G. Dai, "Detecting malware for android platform: An svm-based approach," in Proc. CSCloud, New York, NY, USA, pp. 464–469, 2015.

[27] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket.," in Ndss, vol. 14, pp. 23–26, 2014.

[28] A. Makandar and A. Patrot, "Malware analysis and classification using artificial neural network," in Proc. I-TACT-15, Bangalore, India, pp. 1–6, 2015.

[29] Y. Fan, Y. Ye and L. Chen, "Malicious sequential pattern mining for automatic malware detection," Expert Systems with Applications, vol. 52, pp. 16–25, 2016.

[30] H. Fereidooni, M. Conti, D. Yao and A. Sperduti, "ANASTASIA: ANdroid mAlware detection using STatic analySIs of Applications," in Proc. NTMS, Larnaca, Cyprus, pp. 1–5, 2016.

[31] Y. Aafer, W. Du and H. Yin, "Droidapiminer: Mining api-level features for robust malware detection in android," in Proc. ICST, Sydney, NSW, Australia, pp. 86–103, 2013.

[32] M. Zhang, Y. Duan, H. Yin and Z. Zhao, "Semantics-aware android malware classification using weighted contextual api dependency graphs," in Proc. ACM SIGSAC, Scottsdale Arizona USA, pp. 1105–1116, 2014.

[33] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross and G. Stringhini, "Mamadroid: Detecting android malware by building markov chains of behavioral models," arXiv Prepr. arXiv1612.04433, 2016.

[34] D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee and K. P. Wu, "Droidmat: Android malware detection through manifest and api calls tracing," in Proc. ASIAJCIS, Washington, D.C. USA, pp. 62–69, 2012.

[35] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in Proc. COMPSAC, Tokyo, Japan, pp. 664–669, 2018.

[36] S. Yue, "Imbalanced malware images classification: a CNN based approach," arXiv preprint arXiv:1708.08042, vol. 1708, pp. 1–5, 2017.

[37] S. Poudyal, K. P. Subedi and D. Dasgupta, "A framework for analyzing ransomware using machine learning," in Proc. IEEE SSCI, Bengaluru, India, pp. 1692–1699, 2018.

[38] S. Poudyal, D. Dasgupta, Z. Akhtar and K. Gupta, "A multi-level ransomware detection framework using natural language processing and machine learning," in Proc. MALCON, Fajardo, Puerto Rico, pp. 1-6, 2019.

[39] S. Poudyal, Z. Akhtar, D. Dasgupta and K. D. Gupta, "Malware analytics: review of data mining, machine learning and big data perspectives," in Proc. SSCI, Xiamen, China, pp. 649–656, 2019.

[40] G. E. Dahl, J. W. Stokes, L. Deng and D. Yu, "Large-scale malware classification using random projections and neural networks," in Proc. ICASSP, Vancouver, Canada, pp. 3422–3426, 2013.

[41] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in Proc. MALWARE, Fajardo, PR, USA, pp. 11–20, 2015.

[42] W. Huang and J. W. Stokes, "MtNet: a multi-task neural network for dynamic malware classification," in Proc. DIMVA, Sebastian, Spain, pp. 399–418, 2016.

[43] B. Cakir and E. Dogdu, "Malware classification using deep learning methods," in Proc. ACMSE, Kentucky, USA, pp. 1–5, 2018.

[44] P. Zegzhda, D. Zegzhda, E. Pavlenko and G. Ignatev, "Applying deep learning techniques for Android malware detection," in Proc. ICPS, Cardiff, U. K, pp. 1–8, 2018.

[45] M. Ganesh, P. Pednekar, P. Prabhuswamy, D. S. Nair, Y. Park and H. Jeon, "CNN-based android malware detection," in Proc. ICSSA, Altoona, PA, USA, pp. 60–65, 2017.

[46] T. H. D. Huang and H. Y. Kao, "R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections," in Proc. IEEE BigData, Seattle, USA, pp. 2633–2642, 2018.

[47] L. Shiqi, T. Shengwei, Y. Long, Y. Jiong and S. Hua, "Android malicious code Classification using Deep Belief Network," KSII Transactions on Internet & Information Systems, vol. 12, no. 1, 2018.

[48] Y. S. Yen and H. M. Sun, "An Android mutation malware detection based on deep learning using visualization of importance from codes," Microelectronics Reliability, vol. 93, pp. 109–114, 2019.

[49] Z. Xu, K. Ren, S. Qin and F. Craciun, "CDGDroid: Android malware detection based on deep learning using CFG and DFG," in Proc. ICFEM, Gold Coast, Australia, pp. 177–193, 2018.

[50] A. Pektas and T. Acarman, "Deep learning for effective Android malware detection using API call graph embeddings," Soft Computing, vol. 24, no. 2, pp. 1027–1043, 2020.

[51] L. K. Yan and H. Yin, "Droidscope: Seamlessly reconstructing the {OS} and dalvik semantic views for dynamic android malware analysis," in Proc. Security'12, Washington, USA, pp. 569–584, 2012.

[52] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran and S. Venkatraman, "Robust intelligent malware detection using deep learning," IEEE Access, vol. 7, pp. 46717–46738, 2019.

[53] E. B. Karbab, M. Debbabi, A. Derhab and D. Mouheb, "MalDozer: Automatic framework for android malware detection using deep learning," Digital Investigation, vol. 24, pp. S48--S59, 2018.

[54] M. K. Khan, M. Zakariah, H. Malik, and K. K. R. Choo. "A novel audio forensic data-set for digital multimedia forensics." Australian Journal of Forensic Sciences, vol. 50, no. 5, pp. 525-542, 2018.

[55] "https://contagiominidump.blogspot.ca."

[56] T. Kim, B. Kang, M. Rho, S. Sezer and E. G. Im, "A multimodal deep learning method for android malware detection using various features," IEEE Transactions on Information Forensics and Security, vol. 14, no. 3, pp. 773–788, 2018.

[57] "VirusTotal. Accessed: Sep. 2017. [Online]. Available: https://www. virustotal.com/ko."

[58] "VirusShare. Accessed: Sep. 2017. [Online]. Available: https://virusshare. com."

[59] "Mal-Genome Project. Accessed: Sep. 2017. [Online]. Available: http:// www.Malgenomeproject.org."

[60] "Google Play Store. Accessed: Sep. 2017. [Online]. Available: https:// play.google.com/store."

[61] M. K. Alzaylaee, S. Y. Yerima and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," Computers & Security, vol. 89, pp. 101663, 2020.
[62] "McAfee Labs Threats Predictions Report | McAfee Labs."
[63] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in Proc. IEEE SP, San Francisco, USA, pp. 95–109, 2012.
[64] Y. Li, J. Jang, X. Hu and X. Ou, "Android malware clustering through malicious payload mining," in Proc. RAID, Atlanta, GA, USA, pp. 192–214, 2017.
[65] F. Wei, Y. Li, S. Roy, X. Ou and W. Zhou, "Deep ground truth analysis of current android malware," in Proc. DIMVA, Bonn, Germany, pp. 252–276, 2017.
[66] D. Maiorca, D. Ariu, I. Corona, M. Aresu and G. Giacinto, "Stealth attacks: An extended insight into the obfuscation effects on android malware," Computers & Security, vol. 51, pp. 16–31, 2015.
[67] A. F. A. Kadir, N. Stakhanova and A. A. Ghorbani, "Android botnets: What urls are telling us," in Proc. NSS, Xi'an, China, pp. 78–91, 2015.
[68] L. Li et al., "Androzoo++: Collecting millions of android apps and their metadata for the research community," arXiv Prepr. arXiv1709.05281, 2017.
[69] T. Hastie, R. Tibshirani and J. Friedman, The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media, 2009.
[70] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Communications surveys & tutorials, vol. 18, no. 2, pp. 1153–1176, 2015.
[71] "Ganegedara T. Intuitive guide to convolution neural networks. Towards data science. Available at: https://towards datascience.com/light-on-math-machine-learning-intuitiveguide-to-convolution-neural-networks-e3f054dd5daa (2020, accessed 28 July 2020)."

Table 1. Classical techniques for detecting malware

S. No	Feature	Techniques	Results	Ref
1.	.apk Android executable files	J48 Decision Tree classification	82.7 % accuracy	[20]
2.	ten genuine malware samples	1-Nearest Neighbor classifier	93% of accuracy	[22]
3.	88 features with unrooted device	k-Means, Logistic Regression, Histograms, Decision Tree, Bayesian Networks and Naive Bayes	80% of accuracy	[23]
4.	500 Android .apk files	Random Forest, J48 Decision Tree, and Classification and Regression Tree (CART)	NA	[24]
5.	NA	SVM algorithm	74-83% of accuracy	[25]

Table 2. Artificial neural network techniques for malware detection

S. No	Feature	Dataset	Techniques	Results	Ref
1.	Malware is converted into one-channel gray-scale image	NA	CNN	94.0%	[35]
2.	word2vec technique	NA	gradient boosting algorithm	96%	[43]
3.	API call sequence into RGB images	7192 benign and 24461 malicious samples	CNN	93%	[44]
4.	static method to extract the features	2000 malicious and 500 benign samples	CNN	93%	[45]
5.	converted the .dex files into RGB color codes	2 million benign and malicious Android application	CNN	98.4225%	[46]

Table 3. Convolution neural network techniques for malware detection

S. No	Feature	Dataset	Techniques	Results	Ref
1.	Global wavelet transforms	Mahenur	Artificial Neural Network	96.35%	[28]
2.	Instruction sequence	8847 malicious files and 1460 benign files	All-Nearest-Neighbor (ANN) classifier	97.3%	[29]
3.	Package-level API calls	NA	Artificial Neural Network	99% of the accuracy	[31]
4.	Semantic features	NA	Artificial Neural Network	capable of effectively fighting against malware	[32]
5.	Markov chain	NA	Artificial Neural Network	classify and detect the malware	[33]

Table 4. Datasets for malware detecting the tasks

Dataset	No. of Malware	No. of Benign	Total
Malgenome	1258	37627	38885
Drebin	5555	37627	43182
MalDozer	20089	37627	57716
All	33066	37627	70693

Table 5. Datasets for attributes of task

Dataset	No. of Malware	No. of Family
Malgenome	985	9
Drebin	4661	20
MalDozer	20089	32

Table 6. Different datasets available

S. No	Dataset	Year	Benign ware	Malwa re	Ref
1.	Android Malware Genome Project	2010	863	1260	[63]
2.	DREBIN	2010	123453	5560	[27]
3.	Contagio	2011	-	1150	
4.	Android Malware Dataset	2010	-	24553	[64] [15]
5.	VirusShare	2016	-	65536	[7]
6.	Android PRA Gaurd Dataset	2015	-	10479	[16]
7.	Marvin	2015	50501	7406	[17]
8.	ISCX Android Botnet	2014	-	1929	[18]
9.	Andro Zoo	2018	25000	-	[68]

Table 7. Comparison of different techniques

Methods	Domain	Advantages	Disadvantages
DT	A rule-based tree-structured classification model trained based on information gain of all features in training data	Computational cost is less and easy to implement.	We need to save all the information of the trained model. Space complexity is high.
SVM	It aims to find separating hyperplane in the feature space among its classes so that distance between the hyperplane and its nearest data	Suitable for small sample size but large feature dimensions	Selecting optimal kernel size (k-value) is difficult
ANN	It consists of one or more hidden layers between the input and output layer. Stores input data information as weights in the hidden layer using the backpropagation algorithm	Suitable for pattern recognition problem with high accuracy	Computational complexity is high compared to other algorithms
CNN	The convolution layer of CNN extracts features from training data in a generative fashion using several hidden layers and a pooling layer that pulls that information to predict output	Very useful for image classification and pattern recognition	Computationally complex. Performance degrades with low sample size

Table 8. Future directions and open challenges

S. No	Domain	Open Challenges	Future Directions and Recommendations
1.	Evaluating the performance	Traditional performance metrics are failing for new attacks, domains and environments	• Developing new tools. • Listing the commo criteria. • Establishing open online system for new researchers to contribute in this cause.
2.	Designing the Security	Cybersecurity techniques proposed are failing to handle new threats and degrading the performance when tested with different kind of threats	• Exploring wide range of attacks. • Proposing mathematical security models
3.	Machine learning techniques and its advancements	Traditional ML techniques are failing since the attackers are fooling the models and making it to overfit.	• Develop Advanced Machine Learning techniques with the capability of robust feature extraction techniques and handling overfitting issue
4.	Deep learning model's robustness	The linear nature of DNN-classifier is the main source of the issue.	• Develop more robust DNN models
5.	Preserving the privacy aspects	These techniques could not handle the privacy issued related to datasets as the privacy of the datasets used in the systems are getting leaks	• More research is needing to save the privacy-preserving issue and not effecting the accuracy of the proposed method
6.	Dataset issues	They are lacking in big sizes and also with rich features like fully labelled, structured and complete with diversity and details about the attacks, its domains	• Develop more robust dataset to handle real cybersecurity issues.

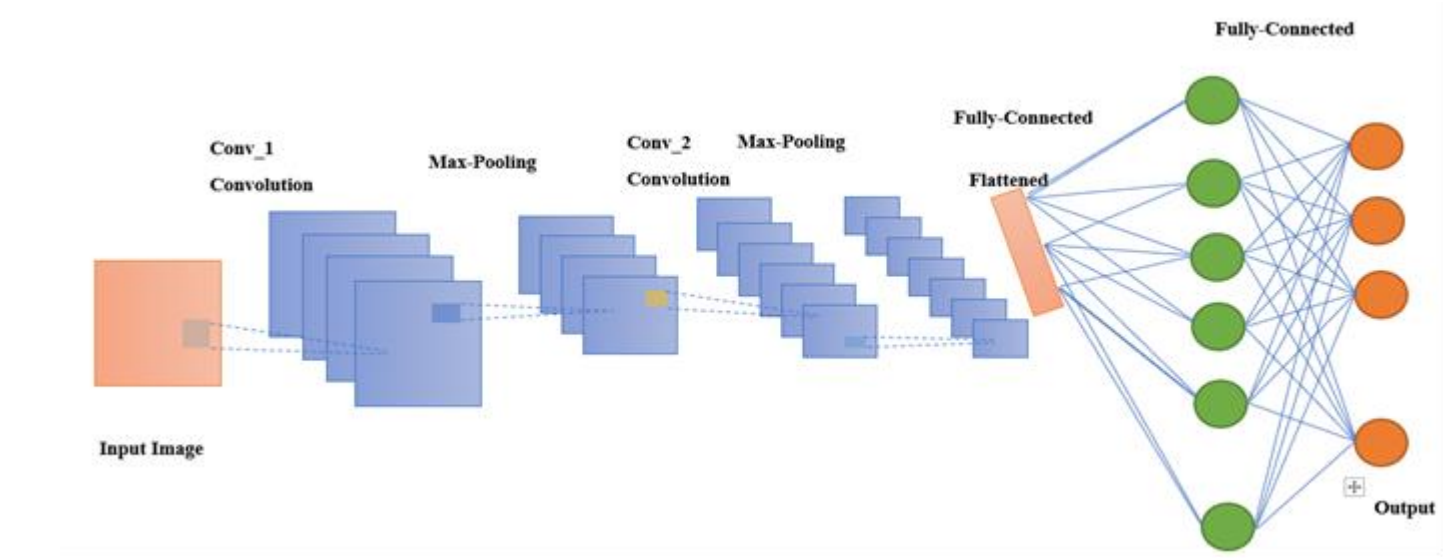

Figure 5. Convolution Neural Network