Deep Image: An Efficient Image-Based Deep Conventional Neural Network Method for Android Malware Detection

Marwa A. Marzouk [1],* and Mohamed Elkholy [2]

[1] Information Technology Department, Matrouh University, Marsa Matrouh, Egypt
[2] Computer Engineering Department, Pharos University in Alexandria, Alexandria, Egypt;
Email: eng_mikholy@alexu.edu.eg (M.E.)
*Correspondence: mabdelazeem@nctu.edu.eg (M.A.M.)

***Abstract*—The continuous increment of malware and its complexity motivated researchers to implement techniques to detect and classify it. Manual detection of malicious files is time consuming and shows poor results. Recently, Deep Convolution Neural Networks (DCNN) shows promising results in malware detection. DCNNs include large number of fully connected layers that are capable to deal with fast iterations of Android malware. Compared to the existing approach, DCNN shows high performance and accuracy in detecting different types of malwares. The proposed work combines Scale-Invariant Feature Transform (SIFT) and DCNN to detect malware features. Combining SIFT with DCNN allow higher accuracy of features classification and overcome the problem of single-feature extraction. The proposed method is compared to existing approaches to malware detection in terms of anticipated time and detection accuracy. The experimental results showed the significant enhancement offered by the proposed work in terms of accuracy and performance.**

***Keywords*—malware, Deep Convolution Neural Networks (DCNN), Scale-Invariant Feature Transform (SIFT), color image transformation**

I. Introduction

Any software that harms a user, a computer, or a network is referred to as malware [1]. According to McAfee Labs Threats Reports, 100,000,000 new malware samples were found in Quarter 1 and Quarter 2 of 2020, but total malware for the same period reached 1,200,000,000 samples [2]. Furthermore, due to the broad nature of cyber-attacks against the android mobile system, the mobile malware detection domain has received significant interest in both academic and commercial sectors. Despite the large number of works completed in this sector, there is a gap between the completed works and the large number of harmful programs launched daily. As a result, malware identification has emerged as one of the most crucial network security jobs for both businesses and individual users. A single assault can cause a data breach and substantial harm [3, 4]. A fast and effective malware detection approach is therefore urgently needed.

Deep learning is recently applied in a variety of academic fields. It shows promising results in image recognition [5]. The proposed wok acquires a malware training data set to identify and categorize malware using deep learning. Data is classified into three categories. The first category is Application Program Interface (API) sequences (for example; Create Process, I/O requests) [6–8]. We can obtain the API sequences by executing malware. However, running malware and obtaining the API sequence takes a long time. The second kind is the opcode sequences [4]. The opcode sequences can be obtained from malware assembly programs (e.g., MOV, ADD). The third form is malware pictures [3, 4], which is the focus of this paper.

The proposed work introduces a framework that detects malware risks. To achieve a complete detection of malware, a method based on deep convolution neural networks and color image visualization is suggested. Malware images have a distinct kind of nature that differs from normal scenes images. A normal scene image has continuous patterns, whereas malware images include patterns that have several undefined shapes. Hence, deep learning algorithms are promising to detect these kinds of patterns. Large significant characteristics of malware binaries can also help with malware family categorization performance. With huge image datasets and features, deep learning gives better results. Deep learning, as opposed to machine learning, may automatically use filters to reduce noise. Deep learning algorithms perform better when utilized with color images. So, for malware detection, we utilize color image transformation. The main contributions of the proposed work are listed below

- A promising design for detecting malware is presented assaults on mobile operating systems.
- A hybrid method concentrating on color image visualization, Scale-Invariant Feature Transform (SIFT), and Deep Convolution Neural Networks (DCNN) is developed that is both affordable and

Manuscript received December 17, 2022; revised February 28, 2023; accepted May 22, 2023; published August 22, 2023.

flexible computationally and has lower run-time requirements.

- A comparison is performed between the suggested approach and earlier malware detection methods. The experiments showed that the recommended approach is more reliable, stable, and economical.

The major parts of the proposed work are structured as follows: Related Works, Methodology, Experimental Results, and finally, Conclusions.

II. Literature Review

To identify malware evasion methods, researchers and anti-malware firms have lately introduced machine learning and deep learning methodologies to the malware detection domain. The detailed related work is covered in the following section. The utilization of byte plot visualization for automatic malware classification was pioneered by Nataraj *et al*. [9]. They retrieved texture-based characteristics from the malware picture after converting all of the samples of malware to visualizations of gray scale byte plots. Gastrointestinal Stromal Tumor (GIST), an abstract representation approach, was utilized to compute texture characteristics from images. Their collection contains 9458 samples of malware. They used global image-based attributes to train a K-Nearest Neighbor model using Euclidean distance as the distance metric to classify malware samples and obtained 97.18% accuracy.

Makandar *et al*. [10] transformed malware into a 2-dimensional grey scale image and then used texture-based characteristics to detect the samples. Utilizing the Mahenhur dataset, which contains 3,131 binary samples from 24 distinct malware categories. In their research, they used the Gabor wavelet transform and GIST to retrieve texture-based global features. They reported an accuracy of 96.35% when detecting malware using Artificial Neural Networks (ANN). Previous research in this field, such as [9, 11–15], employs standard mapping methods to convert malware binaries to images, even though image-based malware categorization is a unique approach that can overcome anti-analysis measures. Therefore, it is possible to ignore the malware's semantics. According to our observations, the more data provided to classifiers, the higher the accuracy rate that can be stored. Kalash and Rochan *et al*. [3] implemented the Multi-scale Convolutional Neural Networks (M-CNN) model, which is based on the VGG-16 image classification architecture [16]. The last layer of an artificial neural network can be replaced using techniques that use an Support Vector Machine (SVM) classifier [17]. To classify imbalanced malware images, Yue [18] developed a weighted softmax loss for CNNs and got an accurate classification. Gilbert and Mateu *et al*. [19] developed a model with three convolutional layers and one fully connected layer, which they tested on the Malimg and Microsoft Malware Classification Challenge datasets. A methodology for malware identification utilizing a Convolution Neural Networks (CNN) that categorized images of malware was proposed by Seon and Kim [20]. They separated their experiments into two groups. When using the top-1 and top-2 ranked values, Malware was first accurately classified into 9 families with an accuracy percentage of 96.2% and 98.4% in the initial round of testing.

Hybrid methods hold great promise because they significantly outperformed static and dynamic methods alone. Santos and Devesa *et al*. [21] proposed a novel approach that combines both static and dynamic information to train a malware classifier. They combined the frequency of operational codes, a static feature, with the implementation trail of an executable, obtained by detecting operations carried out, system calls, and exceptions thrown, a dynamic feature, to apply a hybrid technique that exceeds both approaches when used alone. They tested their strategy by using a range of machine learning algorithms, including decision trees, K-Nearest Neighbors (KNN), Bayesian networks, and SVM, to two different datasets.

Islam and Tian *et al*.'s classification method [22] for separating the binaries into benign and malware files included both dynamic-based and static-based criteria. They made use of API variables and API user-defined functions, as well as printed sting information and frequency of method length. They use 541 benign samples and 2939 malignant samples to test their model. They obtained an accuracy of 97.055% when categorizing malware samples using combined Meta classifiers like SVM, IB1, Decision Tree (DT), and Random Forest (RF). Their outcomes were an upgrade over the earlier ones [23]. The novel method was introduced by using global features of malware visualization and texture patterns for malware classification based on binary texture analysis [14]. To extract effective texture feature vector classification. The advantages of this visualization technique are based on the image processing approach. The file whether it is packed or unpacked can be computed competently which is important for large malware datasets. This technique uses only static analysis that's why it is limited because it does not use dynamic analysis.

As previously stated, various academic research on the subject of Android malware detection has been conducted, some of which will be mentioned in this section. SafeDroid is a static analysis-based approach. has been presented in [24]. The suggested methodology relies on analyzing the DEX (Dalvik Executable) coding to retrieve binary relevant features that were used to learn various machine learning classifiers. A random forest classification model has also been trained by Zhu and You *et al*. [25], several criteria, including authorization, sensitive APIs (application program interfaces), system logs, and access frequency, which may be used to determine if an Android app is harmful or not. Long and Yu [26] proposed a lightweight system based on machine learning that can discriminate between benign and malicious applications. They also collected application characteristics using both static and dynamic approaches. Furthermore, They describe a unique method for reducing the dimensionality of features that have proved successful in feature selection. Huang and Kao [27] suggested a color image-based approach for detecting Android malware. They tested numerous CNN models that had performed well in the ImageNet LargeScale Visual Recognition Challenge (ILSVRC) [28]. Their technique identified malware with an accuracy of 98.42%. However, while creating the photos, they did not take the structure of the DEX file into account. Gennissen and Cavallaro *et al.* [29] suggested an Android malware detection method. They created 10 new types of images by converting images using domain knowledge. They use a transform class CNN model with two layers.dex into a Hilbert curve image with a fractal shape depending on the Dalvik opcode and API details. Their technology is up to 92% accurate. The distinction between this work and the other research stated above is that the data utilized is current. To identify Mobile malware, Suleiman and Sezer *et al.* [30] devised a classifying method based on parallel machine learning. A total of 179 training characteristics were retrieved and separated into relevant API calls and instructions based on genuine malware samples and benign apps that have been built from it: 54 attributes; 125 permissions for the app. A parallel collection of homogeneous classifications, including Simple Logistic, Naive Bayes, Decision Tree, PART, and Ripple Down Rule learner (RIDOR), were used to create a hybrid classifiers.According to their studies, Ripple Down Rule learner outperformed all other classifications, achieving a true-positive rate of 0.95%, a true-negative rate of 0.96%, a false-positive rate of 0.03%, a false-negative rate of 0.04%, and an accuracy rate of 0.96%. Alzaylaee and Yerima *et al.* [31] showed that actual phones are more stable and capable of recognizing more characteristics while analyzing Android applications than emulator environments. Our article compares the detection of Android malware using classical classification techniques vs deep learning methodologies.

III. Materials and Methods

A malware detection model for the Android environment is presented in the propose work as shown in Fig. 1. A prommising learning methodology is introduced to generate more discriminating and robust feature descriptors. The proposed ethodology combines DCNN and SIFT, as well as a color image transformation. Before being processed by a DCNN and SIFT model, the raw Android file is transformed into a color image. The file fingerprints are then compared to a behavioral database to determine if the file is malware or benign. Below is a description of the specific procedure for each section of the proposed design. The three most important modules that make up the proposed architecture's core are color image transformation, SIFT, and the DCNN model.

A. Color Image Transformation

Deep learning algorithms shows better perform in case of using colored images. Hence, the proposed work uses color image transformation. An extractor is used to unzip the.apk file for color image visualization. The .apk file often includes a Class.DEX file with all of the Dalvik binary code. In three phases, we extract binary code from the an.apk file. To begin with, The apk was decompressed. file and obtained the class. Dex file. Second, we use the dex2jar tools to convert the class. Dex file into a Java.Class file [32]. Third, We extract Java binary code. class file by using the JD-GUI decompiler as shown in Fig. 2. Three critical procedures are involved in the translation of malware binary data to color images.

First, substrings are created from the malware's binary bit string. Each substring is 8 bits long and corresponds to a single pixel. Eight bits are thought of as unsigned integers (0–255). Second, a one-dimensional decimal number vector is created from the malware's binary bit string. Third, a color matrix in two dimensions with the required width is created from the one-dimensional decimal numbers vector. based on observations made in practice.

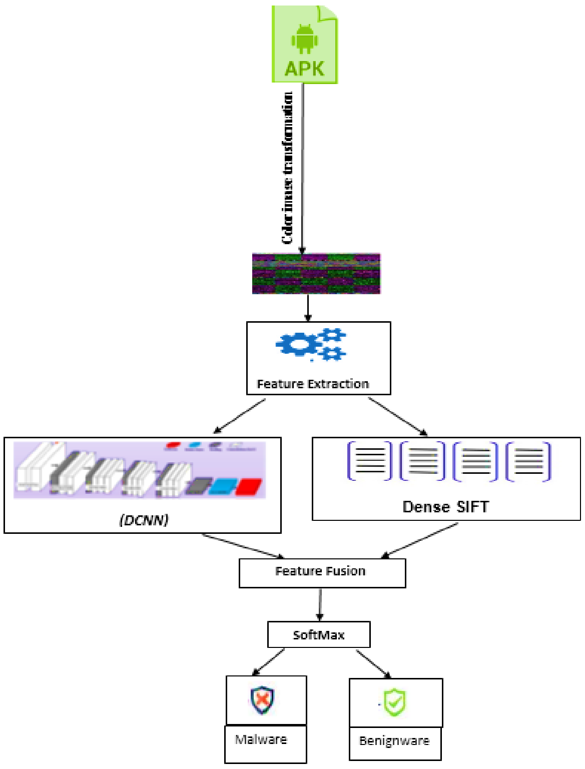
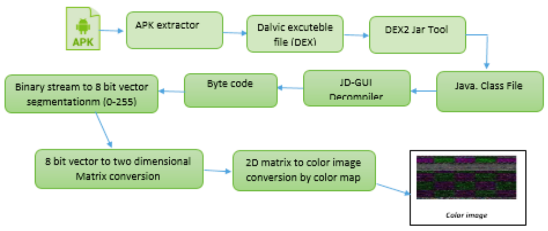


Figure 1. Methodology of malware detection.


APK
APK extractor
Dalvic excuteble file (DEX)
DEX2 Jar Tool
Java. Class File
JD-GUI Decompiler
Byte code
Binary stream to 8 bit vector segmentationm (0-255)
8 bit vector to two dimensional Matrix conversion
2D matrix to color image conversion by color map
Color image
Figure 2. Color image transformation.

B. Dense SIFT

Dense SIFT computes a SIFT descriptor using Lowe's method at each position [33, 34]. It gathers features at each location and scales an image to improve recognition accuracy. It divides an image into small patches and then divides each patch into smaller bins. The feature is then calculated as gradient magnitude histograms in eight different bin orientations. As the sliding window advances, gradient histograms of the image's local neighborhoods are computed. Finally, it uses cascaded connection functions to get the image feature descriptors.

C. Deep Convolutional Neural Network (DCNN)

Throughout this paper, we provide an in-depth examination of the DCNN model as shown in Fig. 3. The suggested DCNN model is composed of four components. The following is a quick description of each layer.

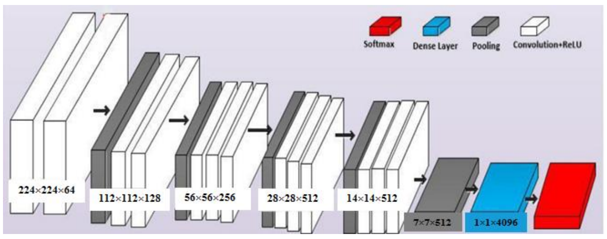


Figure 3. Deep convential neural network [32].

1) Convolution layer

The convolutional layer eliminates noise and improves signal quality initially. The suggested deep learning model's performance is improved by optimizing the convolutional kernel width, the number of hidden units, and the learning rate. The convolutional layer's input consists of many maps [35].

$$x_j^i f\left(\sum_{i \epsilon M_j} x_j^{i-1} \times K_{ij}^i + b_j^i\right) \tag{1}$$

where M_j represents the collection of input maps; K_{ij}^i indicates the convolution kernel that is used to mix the i^{th} input feature map with the j^{th} output feature map; b_j^i displays the bias associated with the i^{th} feature map and f is the activation function.

2) Pooling layer

It improves model performance while reducing the overall feature map dimension numbers. Each sampling result is accompanied by a feature map:

$$x_j^i\, f(down(x_j^{i-1}) + b_j^i \tag{2}$$

where down (.) does a pooling task, and b shows a bias value.

3) Dense layer

It converts a feature vector in two dimensions to a feature vector in one dimension before passing it to an output layer.

4) Output layer

Android examples are classified as either malicious or benign. For data training, the DCNN model applies the Softmax-Cross-Entropy loss.

D. Feature Fusion

To combine the two different kinds of characteristics discussed above, the eigenvector splicing approach is applied in this study. We choose a weighting function to combine the two vectors to create new features since the varied dimensions of the two eigenvectors will result in different proportions. The following is the precise feature fusion process.

1) Integrate the DCNN and SIFT features

The SIFT eigenvector in n dimensions is represented as V_{Li}, where $V_{Li} = (V_{L1}, V_{L2}, \ldots, V_{Ln})$; the m-dimensional DCNN eigenvector is represented as V_{Ci}, where $V_{Ci} = (V_{C1}, V_{C2}, \ldots, V_{Cm})$. A new fused eigenvector $V_{fc} = (V_{N1}, V_{N2}, \ldots, V_{NN})$. The i^{th} element V_{Ni} can be calculated as follows:

$$V_{fc} = \alpha V_{Ci} + (1 - \alpha)\, V_{Li} \tag{3}$$

while α is represent a real value between 0 and 1, This corresponds to the weight of the two sorts of characteristics when combined. If N image samples are given and each sample has a dimension of D, then to determine the value of each class label in M classes, the Softmax function will be applied, as illustrated.

$$P_{yi} = \frac{e^{fn}}{\sum_{j-1}^{M} e^{fn}}\ for\ i = 1, \ldots, m \tag{4}$$

$$f_{yi = W_i^T\, X + b_i} \tag{5}$$

where f_{yi} is the score function and Y_i is the i^{th} linear prediction. f_i is the sum of all sample score function values. Each sample's size X_i is $D \times 1$, W_i^T, i is the i-th weight matrix, the size is $M \times D$. The deviation is b, and the size is $D \times 1$. For an input sample x_i and $j = 1, \ldots, M$, the probability value $P = (Y_i|X_i)$ of a specific class can be represented as a matrix as:

$$\begin{pmatrix} \mathrm{P} = (Y_i|X_i\,; W) \\ \mathrm{P} = (Y_i|X_i\,; W) \\ M \\ \mathrm{P} = (Y_i|X_i\,; W) \end{pmatrix} = \frac{1}{\sum_{j-1}^{M} e_j^T x_i} \begin{pmatrix} e_j^T x_i \\ e_j^T x_i \\ M \\ e_j^T x_i \end{pmatrix} \tag{6}$$

We may construct the loss function using Eqs. (4)–(6) by using the backpropagation strategy based on the stochastic gradient descendant optimization technique to minimize Eq. (7).

$$L = -\log\left(\frac{e^{fn}}{\sum_{j-1}^{M} e^{fn}}\right). \tag{7}$$

2) Dimension reduction of fusion features

Because of the integration of the two types of features, there is a lot of duplicated information, which contributes to the high V_{fc} dimension. As a consequence, we increased the number of nodes in our network model's last fully connected layer to 256, which is comparable to lowering the dimension of V_{Ni} through our model, removing redundant features and inventing new, relevant data. The network model's Softmax layer receives the fusion feature to provide the final classification recognition result.

IV. Result and Discussion

The Drebin Android Malware dataset [36], Malgenom Android Malware dataset [37], and Android Malware Dataset (AMD) [38] were used to test the suggested approach. Three image datasets were created, each with 10,000 samples, including 5000 benign photos and 5000 malware images. The samples of malware include 5000 samples chosen at random from the popular Android malware datasets AMD, Drebin, and Malgenom, as well as 5000 examples chosen at random from the Android malware dataset. The Drebin has 5560 Android malicious applications from 179 families, whereas the Malgenom contains 1260 malicious apps from 49 families. The AMD comprises 24,553 samples classified into 135 different malware families. The benign applications, on the other hand, were downloaded from the Google Play store using the free internet downloader APKPure.Based on scanning the downloaded applications with the Virus Total online API, a Python script has been created to guarantee that they are safe. The DCNN model is built through tests using the Graphics Processing Unit (GPU) version GTX1080 NVIDIA, 758GB of RAM, 64-bit Ubuntu 16.04, and Python Tensorflow 1.9. Hyperparameter-based deep learning model creation is made possible by TensorFlow.The system uses multi-dimensional arrays to carry out operations. To expedite the detection process, parallel implementation is used [39].

Precision, recall, and accuracy were the three assessment variables we utilized to assess performance. Malware samples number classified as true or false were indicated by the number of True Positives (TPs) and False Positives (FPs). The number of True Negatives (TNs) and False Negatives (FNs) indicated how many benign samples were determined to be true or false [40].

$$recall = \frac{TP}{TP+FN} \tag{8}$$

$$percision = \frac{TP}{TP+FP} \tag{9}$$

$$accuracy = \frac{recall+percision}{2} \times 100 \tag{10}$$

Table I displays the comparative results of various Deep Learning (DL) techniques for malware detection. For evaluating the efficacy of the proposed malware detection technique, malware detectors based on pre-trained DL models such as DCNN and their variations are utilized. For the three datasets, the proposed model outperforms DL-based malware detection techniques in terms of performance. The proposed model had an accuracy of 98.38% for the Drebin dataset, 98.83% for the Malgenom dataset, and 99.15% for the AMD dataset.

Table II shows how long it took the proposed model to train and test the data. In terms of computational efficiency, the proposed model is compared against malware detectors based on several DL approaches. When compared to previous malware detection techniques based on deep learning, the studies show that the proposed malware detection model requires less time to test the samples.

TABLE I. A Comparison of the Proposed Approach with Deep Learning-Based Methods for the Three Training Datasets

Models	**Drebin dataset**			**Malgenom Dataset**			**AMD Dataset**		
	Acc %	**Pr**	**Re**	**Acc %**	**Pr**	**Re**	**Acc %**	**Pr**	**Re**
DCNN	97.58	0.977	0.974	95.70	0.96	0.950	94.3	0.9400	0.9420
VGG19	97.50	0.980	0.975	88.80	0.89	0.880	96.3	0.9640	0.9630
Inception-v3	97.70	0.987	0.986	93.00	0.93	0.930	95.0	0.9570	0.9450
Xception	98.00	0.980	0.980	96.80	0.97	0.960	97.5	0.9770	0.9740
DenseNet-121	98.12	0.980	0.980	96.80	0.96	0.970	95.0	0.9540	0.9520
Proposed	98.38	0.986	0.989	98.83	0.98	0.989	99.2	0.9899	0.9912

TABLE II: Compare Malware Detection Algorithms Based on Deep Learning on the Basis of Computing Time

Models	**Training Time (in a sec)**			**Testing Time (in a sec)**		
	Drebin	**Malgenom**	**AMD**	**Drebin**	**Malgenom**	**AMD**
DCNN	6140	4406	10946	7.82	8.14	8.58
VGG19	5174	3652	12721	6.67	6.84	7.04
Inception-v3	5604	4146	11379	5.89	6.08	6.36
Xception	5674	4226	10448	5.08	5.53	6.36
DenseNet-121	6574	5259	8328	8.48	8.70	8.96
Proposed	**1911**	**2199**	**2288**	**3.89**	**4.99**	**4.89**

Table III displays the accuracy, precision, recall rate, and time for each method. We ran tests with three distinct virus image sizes, 224×224, 229×229 and 192×192. Our solution obtained 99.33% detection accuracy, precision, and recall rate, which was higher than existing machine learning and deep learning malware classifications. The result indicates our method’s capacity to detect correspondences between comparable visual components and allows malware analysis to categorize malware and identify variants. Although our technique had a classification time of 58 ms to 4 s with different types of image sizes, which was significantly longer than other methods, it was more accurate in processing small-scale and large-scale malware analysis.

TABLE III. COMPARISON OF DETECTION PERFORMANCE BETWEEN PREVIOUS APPROACHES AND THE SUGGESTED METHOD

Methods	Image size	Acc (%)	Pr	Re	Time
Songqing *et al*. [18]	ND	97.32	ND	ND	ND
Abien *et al*. [41]	ND	84.92	85	85	11ms
Zhihua *et al*. [42]	192 × 192	93.20	93.40	93.00	48ms
Aziz et *al*. [12]	128 × 128	89.11	ND	ND	ND
Proposed Method	192 × 192	99.28	99.33	99.15	58ms
	224 × 224	99.18	99.24	99.12	2 s
	229 × 229	98.68	98.68	98.54	4 s

Table IV represent evaluation of the suggested approach against other malware feature extraction algorithms. Three different algorithms are chosen for comparison with the proposed approach in this study.The first method used Local Binary Patterns (LBP) features of malware images [27], the second method applied GIST features of malware images [43], and the third method implemented local and global features of malware images [44]. The dataset from [45] is utilised to assess the suggested approach. It contains 4000 Android malware samples and 2000 benign samples.

For a test set of 9 families, the best average accuracy of the LBP-SVM algorithm was 90.12%, GIST was 91%, Internet Group Management Protocol (LGMP) was 92.39%, and the suggested technique was 92.86%. As a result, it was established that the suggested method's accuracy was dependent on the size of the training and feature algorithms.

TABLE IV. ACCURACY OF THE SUGGESTED APPROACH AGAINST OTHER MALWARE FEATURE EXTRACTION ALGORITHMS

Number of Training Samples (%)	Accuracy (%)			
	LGMP	GIST	LBP	Proposed
10	86.30	87.61	87.04	86.11
20	88.91	85.35	86.48	89.1
30	89.12	88.42	86.64	89.33
40	90.90	90.33	87.06	90.78
50	90.50	89.02	87.07	91.01
60	90.88	89.52	87.31	91.11
70	91.20	90.30	88.30	91.55
80	92.31	89.05	89.22	92.78
90	92.39	91.00	90.12	92.86

Table V Compars different Deep Learning (DL) techniques for malware detection .The proposed model had an accuracy of 98.38% for the Drebin dataset, 98.83% for the Malgenom dataset, and 99.15% for the AMD dataset. In terms of computational efficiency, the proposed model is compared against malware detectors based on several DL approaches. The proposed malware detection model requires less time to test the samples. Finally, by comparing the detection performance between previous approaches and the suggested method. Although having a classification time of 58 ms to 4s with various image sizes, which was much longer than previous approaches, our approach was more accurate in processing small-scale and large-scale malware analyses.

TABLE V. HYPER-PARAMETERS FOR THE VALUES WHERE THE PROPOSED MODEL GIVES THE BEST RESULTS

Proposed Model Outperforms DL-Based Malware Detection Techniques in the Performance									
Models	**Drebin dataset**			**Malgenom Dataset**			**AMD Dataset**		
	Acc (%)	**Pr**	**Re**	**Acc (%)**	**Pr**	**Re**	**Acc (%)**	**Pr**	**Re**
Proposed	98.38	0.986	0.989	98.83	0.9877	0.9889	99.2	0.9899	0.9912

Proposed Malware Detection Model Requires Less Time to Test and Traing the Samples						
Models	**Training Time (in a sec)**			**Testing Time (in a sec)**		
	Drebin	**Malgenom**	**AMD**	**Drebin**	**Malgenom**	**AMD**
Proposed	**1911**	**2199**	**2288**	**3.89**	**4.99**	**4.89**

Accuracy, Precision, Recall Rate, and Time with Different Image Size					
Methods	**Image size**	**Acc (%)**	**Pr**	**Re**	**Time in (sec)**
Proposed Method	192 × 192	99.28	99.33	99.15	58ms
	224 × 224	99.18	99.24	99.12	2 s
	229 × 229	98.68	98.68	98.54	4 s

V. CONCLUSION

In this study, a visualization-based framework is provided for detecting Android files as benign or malicious. The suggested methodology is based on transforming the contents of some APK archives into color images and detecting malware using image processing methods and deep learning techniques. To extract the malware's image features, the DCNN model and SIFT are used. Finally, the detection accuracy of state-of-the-art approaches is compared to the method presented for demanding data sets. The Drebin Android Malware dataset [31], Malgenom Android Malware dataset [32], and AMD malware datasets were utilized to evaluate the proposed technique. In terms of detection accuracy and computing time, All other state-of-the-art models were surpassed by the proposed model. 98.46% for the Drebin dataset, 98.46% for the Malgenom dataset, and 98.21% for the AMD dataset, all of which are greater than the other approaches tested. The suggested approach correctly identified the majority of the obfuscated malware samples, demonstrating its resistance to malware mitigation techniques. The suggested detection method has good accuracy and time performance that is equivalent to traditional machine learning-based systems. To obtain an ideal solution, we will focus on reducing false negatives in the future.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Marwa A. Marzouk conceived and designed the analysis, analyzed the data, and wrote the paper. Mohamed Elkholy collected the data, contributed data and analysis tools, and performed the analysis. Both of the authors wrote the paper.

REFERENCES

[1] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[2] C. Beek, S. Chandana, T. Dunton, S. Grobman, R. Gupta, T. Holden, *et al.*, "McAfee labs threats report, November 2020," McAfee Labs, 2020.

[3] M. Kalash, M. Rochan, N. Mohammed, N. D. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *Proc. 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018, pp. 1–5.

[4] M. El Kholy and A. Elfatatry, "Intelligent broker a knowledge based approach for semantic web services discovery," in *Proc. 2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, 2015, pp. 39–44.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[6] M. M. Madbouly, M. Elkholy, Y. M. Gharib, and S. M. Darwish, "Predicting stock market trends for japanese candlestick using cloud model," in *Proc. the International Conference on Artificial Intelligence and Computer Vision*, 2020, pp. 628–645.

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

[8] M. Elkholy and M. A. Marzok, "Light weight serverless computing at fog nodes for internet of things systems," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 26, no. 1, pp. 394–403, 2022.

[9] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. the 8th International Symposium on Visualization for Cyber Security*, 2011, pp. 1–7.

[10] A. Makandar and A. Patrot, "Malware analysis and classification using artificial neural network," in *Proc. 2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15)*, 2015, pp. 1–6.

[11] L. Nataraj and B. Manjunath, "Spam: Signal processing to analyze malware [applications corner]," *IEEE Signal Processing Magazine*, vol. 33, pp. 105–117, 2016.

[12] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," in *Proc. 2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)*, 2017, pp. 76–80.

[13] L. Liu and B. Wang, "Malware classification using gray-scale images and ensemble learning," in *Proc. 2016 3rd international conference on systems and informatics (ICSAI)*, 2016, pp. 1018–1022.

[14] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, "Malware analysis using visualized images and entropy graphs," *International Journal of Information Security,* vol. 14, pp. 1–14, 2015.

[15] S. Z. M. Shaid and M. A. Maarof, "Malware behavior image for malware variant identification," in *Proc. 2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, 2014, pp. 238–243.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint, arXiv:1409.1556, 2014.

[17] X.-X. Niu and C. Y. Suen, "A novel hybrid CNN-SVM classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, pp. 1318–1325, 2012.

[18] S. Yue, "Imbalanced malware images classification: A CNN based approach," arXiv preprint, arXiv:1708.08042, 2017.

[19] D. Gibert, C. Mateu, J. Planes, and R. Vicens, "Using convolutional neural networks for classification of malware represented as images," *Journal of Computer Virology and Hacking Techniques*, vol. 15, pp. 15–28, 2019.

[20] S. Seok and H. Kim, "Visualized malware classification based-on convolutional neural network," *Journal of The Korea Institute of Information Security & Cryptology,* vol. 26, pp. 197–208, 2016.

[21] I. Santos, J. Devesa, F. Brezo, J. Nieves, and P. G. Bringas, "Opem: A static-dynamic approach for machine-learning-based malware detection," in *Proc. International Joint Conference CISIS'12-ICEUTE' 12-SOCO' 12 Special Sessions*, 2013, pp. 271–280.

[22] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *Journal of Network and Computer Applications*, vol. 36, pp. 646–656, 2013.

[23] T. Xu, M. Xu, Y. Ren, J. Xu, H. Zhang, and N. Zheng, "A file fragment classification method based on grayscale image," *J. Comput.*, vol. 9, pp. 1863–1870, 2014.

[24] R. Goyal, A. Spognardi, N. Dragoni, and M. Argyriou, "SafeDroid: A distributed malware detection service for Android," in *Proc. 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, 2016, pp. 59–66.

[25] H.-J. Zhu, Z.-H. You, Z.-X. Zhu, W.-L. Shi, X. Chen, and L. Cheng, "DroidDet: Effective and robust detection of android malware using static analysis along with rotation forest model," *Neurocomputing*, vol. 272, pp. 638–646, 2018.

[26] L. Wen and H. Yu, "An Android malware detection system based on machine learning," *AIP Conference Proceedings*, vol. 1864, 020136, 2017.

[27] T. H.-D. Huang and H.-Y. Kao, "R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections," in *Proc. 2018 IEEE International Conference on Big Data*, 2018, pp. 2633–2642.

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.

[29] J. Gennissen, L. Cavallaro, V. Moonsamy, and L. Batina, "Gamut: Sifting through images to detect android malware," Bachelor thesis, Royal Holloway University, London, UK, 2017.

[30] S. Y. Yerima, S. Sezer, and I. Muttik, "Android malware detection using parallel machine learning classifiers," in *Proc. 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, 2014, pp. 37–42.

[31] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Emulator vs real phone: Android malware detection using machine learning," in *Proc. the 3rd ACM on International Workshop on Security and Privacy Analytics*, 2017, pp. 65–72.

[32] H. Naeem, F. Ullah, M. R. Naeem, S. Khalid, D. Vasan, S. Jabbar, *et al.*, "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," *Ad Hoc Networks*, vol. 105, 102154, 2020.

[33] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15908–15919, 2021.

[34] Z. Zhan, G. Zhou, and X. Yang, "A method of hierarchical image retrieval for real-time photogrammetry based on multiple features," *IEEE Access*, vol. 8, pp. 21524–21533, 2020.

[35] R. C. Gonzalez, "Deep convolutional neural networks," *IEEE Signal Processing Magazine*, vol. 35, pp. 79–87, 2018.

[36] 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 *Proc. of the 21th Network and Distributed System Security Symposium (NDSS)*, 2014, pp. 23–26.

[37] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proc. 2012 IEEE Symposium on Security and Privacy*, 2012, pp. 95–109.

[38] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep ground truth analysis of current android malware," in *Proc. International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2017, pp. 252–276.

[39] M. Elkholy, Y. Baghdadi and M. Marzouk, "Snowball framework for web service composition in SOA applications," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 13, no. 1, 2022.

[40] M. J. Awan, O. A. Masood, M. A. Mohammed, A. Yasin, A. M. Zain, R. Damaševičius, *et al.*, "Image-based malware classification using VGG-19 network and spatial convolutional attention," *Electronics*, vol. 10, 2444, 2021.

[41] A. F. Agarap, "Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification," arXiv preprint, arXiv:1801.00318, 2017.

[42] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics,* vol. 14, pp. 3187–3196, 2018.

[43] K. Kosmidis and C. Kalloniatis, "Machine learning and images for malware detection and classification," in *Proc. the 21st Pan-Hellenic Conference on Informatics*, 2017, pp. 1–6.

[44] H. Naeem, B. Guo, M. R. Naeem, and D. Vasan, "Visual malware classification using local and global malicious pattern," *Journal of Computers,* pp. 73–83, 2019.

[45] I. T. Jolliffe, *Principal Component Analysis*, New York, NY: Springer, 2002, doi: 10.1007/b98835