IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 13, 2021, accepted December 29, 2021, date of publication January 4, 2022, date of current version January 7, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3140341

An Automated Vision-Based Deep Learning Model for Efficient Detection of Android Malware Attacks

IMAN ALMOMANI[1,2], (Senior Member, IEEE), AALA ALKHAYER[1], AND WALID EL-SHAFAI[1,3]

[1]Security Engineering Laboratory, Computer Science Department, Prince Sultan University, Riyadh 11586, Saudi Arabia
[2]Computer Science Department, King Abdullah II School of Information Technology, The University of Jordan, Amman 11942, Jordan
[3]Department of Electronics and Electrical Communications Engineering, Faculty of Electronic Engineering, Menoufia University, Menouf 32952, Egypt

Corresponding authors: Iman Almomani (imomani@psu.edu.sa) and Walid El-Shafai (eng.waled.elshafai@gmail.com)

This work was supported by Prince Sultan University, Saudi Arabia, under Grant SEED-CCIS-2021-84.

ABSTRACT Recently, cybersecurity experts and researchers have given special attention to developing cost-effective deep learning (DL)-based algorithms for Android malware detection (AMD) systems. However, the conventional AMD solutions necessitate extensive computations to achieve high accuracy in detecting Android malware apps. Consequently, there is a significant benefit in utilizing convolution neural networks (CNNs) in vision-based AMD applications to quickly and efficiently learn without prior stages of reverse engineering processes. Thus, this paper introduces an efficient and automated vision-based AMD model composed of 16 well-developed and fine-tuned CNN algorithms. This model precludes the need for a pre-designated features extraction process while generating accurate predictions of malware images with minimum cost and high detection speed. Such performance is achieved with colored or grayscale malware images, whether by using balanced or imbalanced datasets. Firstly, the bytecodes of the ''classes.dex'' files extracted from the Android benign and malware apps were converted to color and grayscale visual images before forwarding them to the developed CNN algorithms for classification. Then, the detection efficiency of the proposed AMD model was examined and evaluated using the imbalanced benchmark Leopard Android dataset that composes 14733 samples of malware apps and 2486 samples of benign apps. Finally, different experimental scenarios were conducted using balanced and imbalanced Android samples of color and grayscale images generated from the Leopard dataset; to extensively and sufficiently validate the detection and classification performance of the suggested model. Comprehensive assessment classification parameters in the evaluation experiments were applied to prove the high capability of the developed fine-tuned CNN algorithms in recognizing Android malware attacks with low computational overhead. As a result, the detection accuracy reached 99.40% for balanced samples and 98.05% for imbalanced samples. Furthermore, the proposed AMD model outperforms the existing approaches that utilize conventional vision-based algorithms and are tested on the same benchmark Android dataset.

INDEX TERMS Cyberattacks, android, malware detection, visualization, color and grayscale images, imbalanced datasets, deep learning, machine learning, convolution neural network (CNN), fine-tuning, transfer learning.

I. INTRODUCTION

Android Operating System (OS) is dominating the smartphone marketplace holding 72.84% of the mobile market share [1]. Furthermore, due to the open-source nature of the Android platform, users can download applications from several markets such as Google Play Store or third-party marketplace. However, the open nature of Android along with its popularity rose the attraction of malware attackers. Any application with lousy intention is malicious software (malware). Malware is developed to control the user's device, steal his or her information, and interrupt the OS functionality.

The associate editor coordinating the review of this manuscript and approving it for publication was Shuihua Wang.

Android malware can be categorized into several types such as Riskware, SMS, Adware, and Banking [2]. To evade malware detection systems, malicious software developers usually imply small modifications on the original source code of the malware app to generate new malicious software variants. In consequence, identifying the new malicious software variants becomes challenging even if they belong to the same family [3].

In order to overcome the aforementioned challenge, a model can be trained using machine learning (ML) to identify malicious software families in regards to the source code variants efficiently [4]. ML has been deployed in developing malware detection systems using different approaches such as static, dynamic, or hybrid analysis [5]–[7]. In static analysis, the original source code of the Android application is parsed without executing the app. On the other hand, the dynamic analysis studies the app's features and its behaviour during the run-time. However, in both approaches, retrieving the features of the Android Application Package (APK) by reverse engineering or run-time execution consumes processing time and computational resources. However, a model can be easily trained utilizing the deep learning (DL) approach by converting the malicious classification issue to an image classification issue [8]–[11].

Convolutional Neural Networks (CNN) is a type of deep learning implemented in a multi-layer algorithm to sufficiently classify a large set of images. Inspired by the effective classification of CNN, this paper proposes an automated vision-based DL model for Android malware detection (AMD) systems. The substantial contributions of this work are:

- Presenting a comprehensive review of the static-based, dynamic-based, and vision-based AMD approaches.
- Introducing an automated vision-based AMD model for accurate and efficient detection of malware attacks existing in the Android operating system.
- Developing 16 different fine-tuned DL-based CNN algorithms (Xception, VGG16, VGG19, DarkNet53, MobileNetV2, ResNet101, AlexNet, ResNet50, ResNet 18, InceptionV3, DarkNet19, ShuffleNet, Places365-GoogleNet, NasNetMobile, GoogleNet, and Squeeze Net) to proficiently classify benign apps from malware apps without the need for extensive computations of reverse engineering or features extraction stages.
- Testing two binary classification scenarios using imbalanced (14733 malware samples and 2486 benign samples) and balanced (2486 malware samples and 2486 benign samples) Android apps datasets; to demonstrate the success of the developed fine-tuned CNN algorithms to work on different balanced and imbalanced datasets sizes without the need for data augmentation techniques like other conventional classification approaches.
- Accomplishing lower computational overhead and higher detection accuracy for the proposed vision-based automated AMD model compared to conventional detection models. This is achieved with fewer training iterations by using only the fine-tuning process of the CNN layers, hyperparameters, and CNN optimization techniques.
- Performing extensive experiments using both visual color and grayscale images of Android benign and malware apps to precisely evaluate the detection performance of the suggested AMD model even when it is applied on different visual image representations.
- Executing comprehensive simulation tests to prove the validity and efficiency of the proposed automated AMD model using 16 various detection and classification parameters.
- Implementing different experiments to check the storage capacity and complexity performance of the developed CNN algorithms to prove the simplicity and efficiency of the proposed automated AMD model in recognizing Android malware attacks.
- Conducting a comparative study in terms of the obtained classification accuracy of Android malware attacks; to confirm the superiority of the proposed AMD model in comparison to recent related and conventional AMD models.

The rest of the paper is structured as follows. Section II discusses a background on the Android application package and related works. Section III introduces the proposed automated vision-based AMD model. Section IV displays the experimental results and discussions. Finally, Section V concludes this paper and presents possible future work.

II. BACKGROUND AND LITERATURE REVIEW

A. ANDROID APPLICATION PACKAGE

Android Application Package, APK, is a zipped file for distributing and installing applications by the Android OS [12]. However, unzipping the APK file results in mainly retrieving the following:

- *AndroidManifest.xml*: a binary XML file format that contains metadata of the app such as app name, permissions, and version.
- *classes.dex*: a Dex file format that contains the app code.
- *resources.arsc*: a file that contains the pre-compiled resources of the app, such as styles, colors, and strings.
- *assets*: a directory that contains the app assets.
- *res*: a directory that contains all the app resources which are not included in the resources.arsc file.
- *lib*: a directory that contains all the app libraries.
- *META-INF*: a directory that contains the metadata of the APK, such as the APK signature.

A further step can be implemented by utilizing some reverse-engineering tools to get different formats of the .dex file as shown in Fig. 1. For example, the app classes can be retrieved in .smali format by using APKtool. Furthermore, the classes in Java format can be restored from the .dex file by deploying Dex2jar tool [13]. This reverse engineering step might be necessary for some research works to implement deep feature extraction [14]–[16].

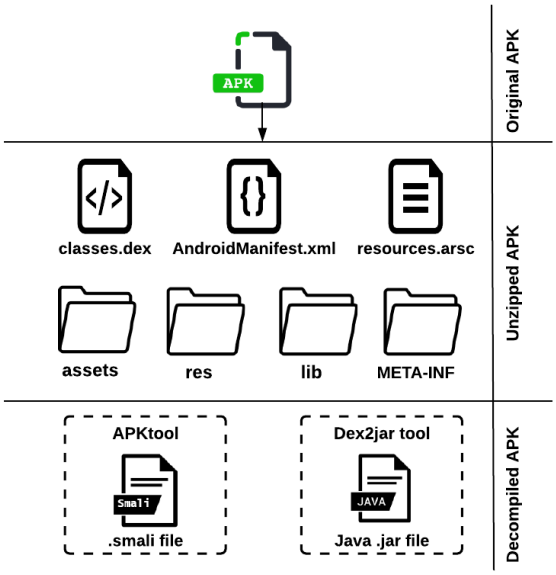


FIGURE 1. Android APK decompilation process.

B. STATIC/DYNAMIC ML-BASED ANALYSIS

Machine learning (ML) is an approach in which the system learns a pattern, develops a model, and generates predictions by observing only the input data. In implementing malicious detection systems for Android applications, the machine learning analysis approach utilizes several features of the Android app including API calls, permissions, and control flows [17]. However, obtaining such features can be performed by static, dynamic, or hybrid approaches [18]. In the static analysis approach, the source code of the Android malware is parsed without executing the application. Several static features can be extracted by scanning the binary files of the malicious software including permissions and API calls [19]. Subsequently, various machine learning algorithms can be deployed in the classification process. Despite the fact that the static analysis is inexpensive, it lacks to detect zero-day and obfuscated malware attacks [20].

Besides static-based analysis applications, there are various contributions to dynamic-based malware analysis [21]–[24]. The dynamic analysis approach observes the behavioral features of the malware by running the Android APK. To avoid any damage on real devices, the execution is performed in isolated virtual machines. Various behavior-based characteristics can be acquired during the dynamic analysis such as network traffic activities [25], API calls [26], and system log files [27]. For further classification, the obtained behavioral features are combined in the features dataset [17]. However, since the dynamic analysis is performed in an isolated environment, the malware may change its behavior during its run-time. Therefore, the dynamic analysis may be insufficient in capturing the real behavior of the malware attack.

C. VISION ML-BASED ANALYSIS

The research work on the visual-based analysis approach offered a new direction to deploy convolutional neural network (CNN) algorithms to effectively detect malicious software. Huang *et al.* developed R2-D2, a color-based CNN detection system for the Android platform [8]. The system converted the classes.dex file of the Android application into an RGB (Red, Green, Blue) image. Subsequently, the colored image was utilized in the feature extraction and training of the CNN model. The research work results in creating a database namely, Leopard Mobile database.

Several papers have utilized Leopard Mobile database [9]–[11], [28]. The author of [9] developed a malware threat hunting system (MTHS) using deep CNN (DCNN) and ML. MTHS aims to detect malware by applying machine learning and deep learning on the converted binary files of malware applications. However, the proposed system was trained on colored images only. Another system, TensorFlow, was proposed by [10]. Initially, the malware source code was filtered then deep learning algorithm was applied to identify the source code plagiarism. In [11], an visual-based malware detection framework was implemented using three Fine-tuned CNN models including InceptionV3, ResNet50, and VGG16. An accuracy of 97.35% was obtained. Nevertheless, the framework endured additional complexity due to the applied augmentation techniques to handle the imbalance samples distribution. Furthermore, Naeem *et al.* developed an industrial Internet of Things malicious software detection scheme utilizing a visual-based CNN algorithm [28]. They proved that malware detection based on colored visualization outperforms the utilization of gray-scale images.

However, in some works, the malware and benign samples were obtained as APK files from several sources such as Drebin, AMD, and Google Play Store [29], [30]. Subsequently, the APK samples were converted into images that match the CNN model requirements. In [29], the authors created an adjacency matrix of Android APK and converted it into an image as an input to the CNN model. Darwaish *et al.* developed an intelligent mapping algorithm of APK files to RGB images [31]. Their proposed system mapped the Manifest file to the green channel. Furthermore, API calls and opcodes were mapped to the red channel. Finally, the malicious behaviors were mapped to the blue channel. A further investigation has been implemented by [32] utilizing the Drebin database. In the proposed system the malware is detected by identifying maliciously opcode sequence locations in the Android app. However, machine learning algorithms can be combined with the CNN model to enhance the detection performance. In [33], they have substituted the softmax layer of CNN with Support Vector Machine (SVM). The results showed that the fine-tuned CNN-SVM model surpassed the original CNN.

D. RELATED WORK COMPARISON

Table 1 presents a summary and comparison among current works on vision-based Android malware detection. It is concluded from Table 1 that most of the conventional DL-based or ML-based AMD models in the literature have accomplished certain detection accuracy levels that are not

TABLE 1. Summary and comparison among current works on vision-based Android malware detection.

Related work	Year	Purpose	Dataset	Distribution	Samples (Malware, Benign)	Color Scheme	Input	Approach	Models	Accuracy
[8]	2018	To identify the connection between the malware APK execution and its function calls order.	Leopard	Imbalanced	14,733 M, 2486 B	Colored	classes.dex	CNN	Inception-v3	93%
[9]	2019	To classify malicious software in IoT devices by applying machine learning and deep learning.	Malimg, Leopard	Imbalanced	14,733 M, 2486 B	Colored	APK, classes.dex	ML, DCNN	Scratch model	97.35%
[10]	2019	To classify malicious software by utilizing deep learning techniques to identify source code plagiarism.	Leopard	Imbalanced	14,733 M, 2486 B	Colored	classes.dex	DCNN	Scratch model	97.46%
[11]	2020	To classify malware families by implementing a CNN detection framework utilizing colored images.	Malimg, Leopard	Imbalanced	14,733 M, 2486 B	Colored	APK, classes.dex	CNN	VGG16, ResNet50, InceptionV3	97.35%
[28]	2020	To develop an industrial Internet of things malicious software detection scheme utilizing visual-based CNN approach.	Malimg, Leopard	Imbalanced	14,733 M, 2486 B	Colored	APK, classes.dex	DCNN	VGG-16	97.81%
[30]	2020	To deploy high order feature maps in enhancing the accuracy of malicious software classification.	Drebin, Anzhi	Imbalanced	3962 M, 1000 B	Gray-scale	classes.dex	CNN	Scratch model	95.1%
[31]	2020	To develop an intelligent mapping algorithm of APK files to RGB images.	AndroZoo	Balanced	16000 M 16000 B	Colored	classes.dex, Manifest	ML, CNN	ResNet	99.37%
[29]	2021	To create an adjacency matrix of Android APK to be fed to the CNN model aiming to classify malware families.	Drebin, AMD, Google Play Store	Imbalanced	17906 M, 4460 B	Gray-scale	classes.dex	CNN	Scratch model	98.26%
[32]	2021	To detect malware by identifying maliciously opcode sequence locations in the Android app.	Drebin	Balanced	5560 M, 5560 B	Not mentioned	APK	CNN	Scratch model	98%
[33]	2021	To substitute the SVM algorithm with softmax layer of CNN in order to perform a deep feature extraction of Android application.	Drebin	Imbalanced	5560 M	Gray-scale	classes.dex, resource, manifest, certificate	ML, CNN	CNN-SVM	92.59%
This work		**To design an efficient and automated vision-based AMD model that composed 16 different well-developed and fine-tuned CNN architectures.**		**Imbalanced**	**14733 M, 2486 B**	**Colored**	**classes.dex**	**Fine-tuned CNN**		**98.05%**
			Leopard	**Imbalanced**	**14733 M, 2486 B**	**Gray-scale**			**16 different CNN models**	**97.93%**
				Balanced	**2486 M, 2486 B**	**Colored**				**99.40%**
				Balanced	**2486 M, 2486 B**	**Gray-scale**				**99.20%**

highly appreciated and recommended for efficient Android malware identification in cybersecurity applications. Furthermore, some of them require features engineering steps before performing the learning process. In addition, the conventional detection models used datasets with a small number of samples in the training process that have dramatically reduced the detection efficiency. Thus, because the number of malware apps is increasing considerably and daily, an automated and accurate vision-based AMD model is introduced in this article; to accurately and efficiently detect Android malware attacks. The suggested AMD model composes 16 different CNN algorithms that have been fine-tuned efficiently and adequately to achieve high malware detection accuracy and low malware misclassification.

Consequently, the fine-tuned and developed CNN algorithms suggested for the vision-based AMD process in this

paper are different from conventional AMD models that introduce additional steps for extracting features. In the proposed AMD model, the bytecodes of the benign and malware APKs were converted to color and grayscale visual images before resizing and forwarding them to the developed CNN algorithms to classify them. The goal of transforming and resizing benign and malware apps to graphical images is to generate an Android dataset in a proper structure adapted to the input format and size of the utilized CNN algorithms. The main advantage of using the pre-trained CNN algorithms in the proposed AMD model that they were well-trained previously on more than 14 million digital images of many different classes of the ImageNet database [34]. So, in the proposed AMD model, the transfer learning concept was exploited by employing the already trained features and the obtained optimal weights of the pre-trained CNN algorithms for detecting malware attacks efficiently. This terrific benefit of transfer learning is recommended in AMD tasks, especially when examining and analyzing the performance of malware detection models on imbalanced Android datasets. Moreover, the fine-tuning of weights and hyperparameters of the CNN layers significantly improved the operation of the utilized pre-trained CNN algorithms. Consequently, increasing the detection performance of the proposed AMD model without using reverse-engineering tools or signal processing-based augmentation algorithms.

III. PROPOSED AUTOMATED VISION-BASED AMD MODEL

In the last years, it has been evident that the number of Android malware cyberattacks has increased gradually. As a result, cybersecurity scholars and experts are interested in developing cost-effective and reliable solutions to mitigate the severe impact of such attacks. Therefore, this paper proposes an accurate and automated vision-based Android malware detection (AMD) model that deals with this critical cybersecurity challenge that cannot be neglected. This model composes different fine-tuned DL-based CNN algorithms developed and exploited to detect malware attacks in Android OS efficiently.

The proposed vision-based AMD model is different from the conventional and existing AMD solutions. So, in contrast to the preceding static-based or dynamic-based AMD solutions that necessitate manual procedures for features extraction and collection, the proposed AMD model in this paper can efficiently detect Android malware attacks without extensive computations resulting from extracting many complex features from the analyzed Android apps. To be more specific, as indicated in Fig. 2, the proposed AMD model composed 16 different well-developed and fine-tuned CNN algorithms that preclude the need for pre-designated extracted features. Thus, the proposed AMD model can quickly learn and efficiently differentiate and recognize Android malware and benign apps more accurately.

The main steps of the proposed automated vision-based AMD model are demonstrated in Fig. 2. It comprises three different main modules: (1) Pre-processing module, (2) Training, fine-tuning, and classification module, and (3) Detection evaluation module. The explanations and discussions of these three modules are as follows:

A. PRE-PROCESSING MODULE

In the proposed AMD model, the bytecodes of the classes.dex files obtained from the Android dataset of benign and malware apps have been converted into the three-channels format of visual color images (Red, Green, Blue). Because the type of image files affects the performance of the Android malware detection system, consequently, the classes.dex files of the Android APK files are converted to ''.png'' format images files since it is the most effective file type compared to other image formats. Furthermore, the ''.png'' format is better than other image formats regarding preserving the information included in the image file. The main objective of transforming Android apps into visual images is to acquire more additional features and extra texture details that cannot be obtained and extracted from the original benign and malware apps in their binary formats. So, the Android dataset conversion to visual images avoids the need for reverse feature engineering steps or any specific domain knowledge, as the case in the existing conventional signature-based (static-based) or behavior-based (dynamic-based) Android analysis techniques.

In the conversion process, each 8-bits (bytecode) in the classes.dex file is transformed into an RGB pixel. This process was repeated for all binary bits in the .dex file of all benign and malware apps in the Android dataset. After that, all obtained RGB pixels were accumulated and reformatted to generate the final 2D color image of each Android app (benign or malware).

To precisely evaluate the detection performance of the suggested AMD model on successfully working on different image visualizations and representations, the visual grayscale images of Android benign and malware apps have also generated. Fig. 3 presents samples of the generated color and grayscale images of the benign and malware Android APKs in the Leopard mobile dataset. As shown in Fig. 3, the resulting color or grayscale images have various resolutions with different widths based on the size of their original .dex files extracted from the benign and malware APKs. Table 2 shows the relation between the Android app sizes and the specific widths of the generated visual images.

Furthermore, it is demonstrated from the obtained visual color or grayscale images presented in Fig. 3 that the generated images have various layouts, styles, and forms. So, the malware images have particular visual similarities and attributes that are entirely dissimilar from those of benign images, where each category of them has various distinctive stripes. These remarkable differences in the visualization features of the acquired benign and malware images inspired us to adapt and exploit the common DL-based pre-learned CNN algorithms for AMD challenges and mobile cybersecurity applications. Therefore, these CNN algorithms utilized for general image processing applications of detection,

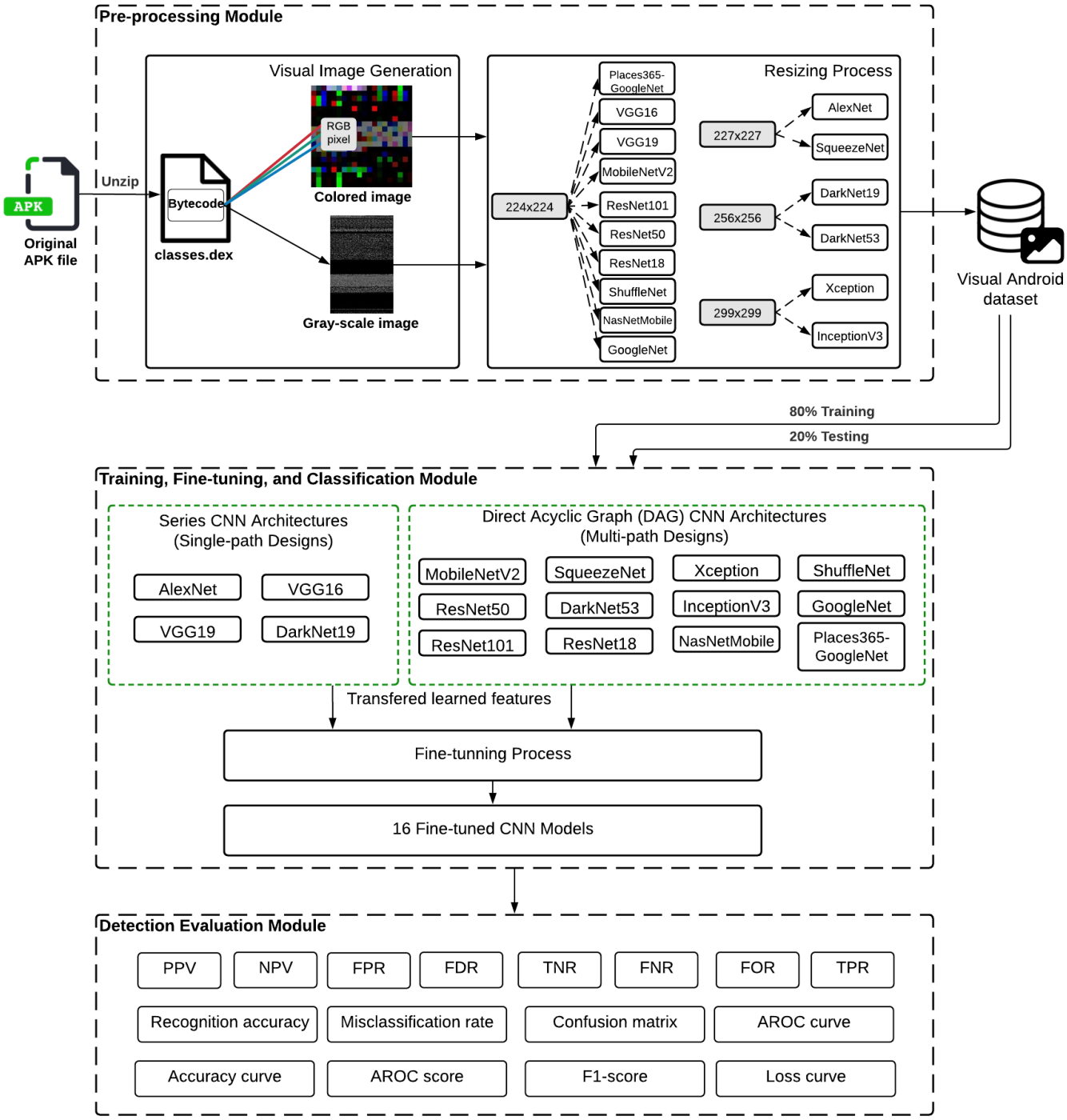


FIGURE 2. The proposed vision-based AMD model.

classification, and recognition tasks have been exploited in the proposed work to detect malware attacks in Android OS.

After obtaining the visual color and grayscale images, they were resized before redirecting them to the suggested fine-tuned CNN algorithms for automated features extraction, training, and classification purposes. The resizing process for the generated Android benign or malware images is a mandatory step where each one of the employed CNN algorithm has its specific resolution for the input image size, as depicted in Table 3.

Additionally, the obtained visual Android dataset of benign and malware images was distributed into two different percentages for testing and training objectives. More simulation experiments were carried out to decide the optimal

TABLE 2. The relation between Android APK sizes and the generated visual image widths.

APK size	Image width
< 10KB	32
10 KB 30KB	64
30 KB 60KB	128
60 KB 100KB	256
100 KB 200KB	384
200 KB 500KB	512
500 KB 1000KB	768
> 1000KB	1024

percentages that can be utilized to achieve efficient malware detection with high recognition accuracy and fast training. The simulations' outcomes disclosed that earmarking 80% of the visual dataset for the training process and 20% of the visual dataset for the testing process have realized the recommended and superior AMD performance compared to the other testing and training percentages for the examined fine-tuned CNN algorithms. Thus, in our work, we did not use the validation set because we used transfer learning CNN algorithms, not CNN algorithms developed from scratch. This is the common practice used in the literature work in the case of using transfer learning CNN algorithms. So, in the case of using transfer learning CNN algorithms, the validation set and test set are combined, and they are considered the testing ratio of the utilized dataset. Therefore, in the experimental analysis, both 80% and 20% of the visual images were chosen randomly by the suggested AMD model.

B. TRAINING, FINE-TUNING, AND CLASSIFICATION MODULE

Most of the existing DL-based AMD algorithms trained malware detection models on Android datasets with a limited number of APKs. Thus, these conventional AMD algorithms have significant malware detection and classification problems because they were not well trained. Consequently, they can not efficiently discriminate the behaviors of benign apps from malware apps due to the limited number of tested samples.

Consequently, the dataset size used for training the CNN models has significant impacts on the detection efficacy, classification accuracy, and the number of computations of the training and testing processes. So, the DL-based transfer learning CNN models are efficient solutions for malware attacks analysis and AMD applications, mainly when the examined Android dataset contains a small number of benign and malware samples as in the benchmark Leopard mobile dataset used in this paper.

Therefore, the proposed AMD model has exploited and employed transfer learning. The already learned features, weights, and hyperparameters of previously pre-learned CNN algorithms tested for image recognition challenges with general images dataset were transferred to the proposed malware image recognition challenge that used a different Android images dataset. Hence, transfer learning was an effective solution for malware detection analysis in the proposed AMD model. The tested Android dataset has a limited and imbalanced number of benign and malware images. Thus, this incredibly avoided the occurrence of over-fitting as possible during the training and testing processes while validating the suggested model performance.

TABLE 3. The image sizes of the CNN algorithms.

No.	CNN	Size
1	Xception	299×299
2	VGG16	224×224
3	VGG19	224×224
4	DarkNet53	256×256
5	MobileNetV2	224×224
6	ResNet101	224×224
7	AlexNet	227×227
8	ResNet50	224×224
9	ResNet18	224×224
10	InceptionV3	299×299
11	DarkNet19	256×256
12	ShuffleNet	224×224
13	Places365-GoogleNet	224×224
14	NasNetMobile	224×224
15	GoogleNet	224×224
16	SqueezeNet	227×227

Different DL-based pre-trained CNN algorithms were previously trained on various natural images such as Xception [35], VGG16 [36], VGG19 [37], DarkNet-53 [38], MobileNet-V2 [39], ResNet101 [40], AlexNet [41], ResNet-50 [42], ResNet18 [43], InceptionV3 [44], DarkNet19 [45], ShuffleNet [46], Places365-GoogleNet [47], NasNetMobile [48], GoogleNet [49], and SqueezeNet [50]. In the proposed vision-based AMD model, the fine-tuned versions of these sixteen CNN algorithms have employed to extract and obtain the significant texture features of the Android malware and benign images. These algorithms were pre-learned and pre-trained on the ImageNet dataset [34] to distinguish different types of visual objects. Consequently, these CNN algorithms can be exploited and re-trained quickly using Android benign and malware images to extract their main visible details and texture features; this is the terrific advantage of the transfer learning process. Therefore, the transfer learned-based CNN algorithms were employed in the proposed model to detect Android malware attacks. They offered effective detection performance through knowledge transfer from general image detection and classification challenges to Android malware image detection and classification challenge studied in this paper.

Therefore, the optimized and fine-tuned versions of two different categories of pre-trained CNN algorithms were utilized in the proposed AMD model: series CNN algorithms and Direct Acyclic Graph (DAG) CNN algorithms. In each one of the series CNN algorithms like AlexNet, VGG16, VGG19, and DarkNet19, the deep CNN layers are organized one after the other. In addition, each series CNN architecture has a single output layer and a single input layer. So, the series CNN algorithms are considered single-path deep CNN designs with no parallel paths of convolutional layers. On the other hand, the DAG CNN

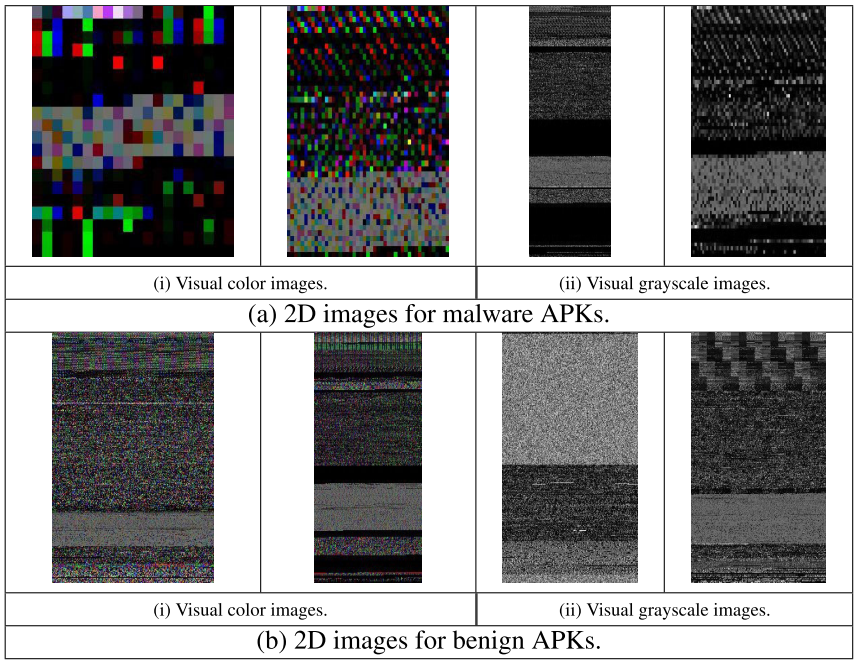


FIGURE 3. 2D visual samples for malware and benign APKs.

algorithms are considered multi-path deep CNN designs, where they have concatenated parallel multi-paths of numerous convolutional layers with different filter numbers/sizes. Thus, the DAG CNN algorithms have deep CNN layers organized by a directed acyclic graph; therefore, they have more complex structures than series CNN algorithms. In addition, each DAG architecture has inputs from different CNN layers and outputs to various CNN layers. The Xception, DarkNet53, MobileNetV2, ResNet101, ResNet50, ResNet18, InceptionV3, ShuffleNet, Places365-GoogleNet, NasNetMobile, GoogleNet, and SqueezeNet are different examples of DAG CNN algorithms. In terms of detection accuracy, the DAG CNN algorithms have higher detection and classification accomplishment than the series CNN algorithms because they can extract more informative and texture features in the training process from the input malware and benign images.

Among the employed CNN algorithms tested by the proposed AMD model, the fine-tuned Xception CNN algorithm achieves the most outstanding and superior detection results for visual Android benign and malware classification compared to other CNN algorithms. Consequently, this paper discusses in-depth details and insights into its structure, training behavior, fine-tuning and optimization hyperparameters, and accomplishment detection outcomes. Thus, the proposed vision-based AMD model has implemented and utilized the fine-tuned structure of the pre-trained Xception CNN algorithm shown in Fig. 4; to classify and detect visualized images of Android malware and benign apps. The DL-based Xception CNN algorithm is previously trained on more general digital images (approximately 14 million images with 1000 different classes) of the ImageNet database [34]. So, the already pre-trained features have exploited and transferred in the proposed AMD model to quickly and accurately detect Android malware attacks.

The Xception CNN algorithm is a modern and enhanced version of the InceptionV3 CNN algorithm [44]. The Xception CNN algorithm is called 'Extreme Inception' algorithm, where the Xception algorithm has the same Inception algorithm by replacing more of the standard convolutional (Conv.) layers with SeparableConv. layers. The SeparableConv. layers are utilized instead on the Conv. layers to factorize the convolution kernel into two smaller kernels. So, the detection and classification performance of the Xception algorithm outperforms that of the InceptionV3 algorithm through proper and efficient use of the algorithm hyperparameters while using a small number of training iterations. The complete structure with the full specifications of the fine-tuned Xception algorithm utilized in the proposed vision-based AMD model is given in Fig. 4.

The input layer of the Xception CNN algorithm has an input image resolution of $299 \times 299 \times 3$. Therefore, before forwarding the visual begin and malware images to the Xception CNN algorithm, they must be resized to $299 \times 299 \times 3$ to meet the proper input size of the input layer. As shown in Fig. 4, the visual malware and benign images are firstly forwarded to the entry flow. Then, the resulting feature maps pass to the middle flow, repeated eight times. Finally, the resulting feature maps go through the exit flow.

The Xception CNN algorithm consists of 36 Conv. and SeparableConv. layers used for extracting the main informative texture features from the input visual malware and benign

images. These 36 stacked Conv. layers are structured and arranged into 14 separable modules (blocks). These modules have linear residual networks except for the first and last modules. In the proposed Xception CNN algorithm, only one fully-connected layer is utilized before the final softmax layer used for detection and classification purposes. Thus, the Xception CNN algorithm composes a linear group of SeparableConv. layers, including more linear residual connections. In the Xception CNN algorithm, all Conv. and SeparableConv. layers are followed by batch normalization layers that are not incorporated in Fig. 4 for simplicity in the presentation. In addition, all SeparableConv. layers utilize a depth multiplier of 1, not a depth expansion. The stride value of 2×2 is used for all Conv. and MaxPooling layers. The ReLU activation function is used to accelerate the training process. Also, the Xception CNN algorithm composes one GlobalAveragePooling layer and four MaxPooling layers with a kernel value of 3×3. The objective of the MaxPooling (Maximum Pooling) layer is to estimate the maximum value for every patch of the feature map, while the GlobalAveragePooling layer estimates the average value for every patch on the feature map.

The most important advantage of the fine-tuned Xception CNN algorithm compared to other CNN algorithms that it can be improved easily where its stacked modules have internal repeated types of layers that can be simply adapted and modified. In addition, the fine-tuned Xception algorithm improves the detection performance without the need to perform deeper training, where the composed Conv. or SeparableConv. layers have different kernels that can discover and learn distinctive texture features in the benign and malware images with a small number of training iterations. Therefore, it is computationally efficient and attractive to be employed for detecting Android malware attacks. Further information and explanations of the rest of the other utilized 15 different pre-trained CNN algorithms (VGG16, VGG19, DarkNet-53, MobileNet-V2, ResNet101, AlexNet, ResNet-50, ResNet18, InceptionV3, DarkNet19, ShuffleNet, Places365-GoogleNet, NasNetMobile, GoogleNet, and SqueezeNet), could be investigated and explored in [36]–[50].

Besides exploiting the advantages of transfer learning in the proposed AMD model, the whole hyperparameters of the employed CNN algorithms are fine-tuned. So, the proposed AMD model utilized fine-tuning, not other types of tuning like shallow tuning or deep tuning [51]. This is because fine-tuning is better than these tuning types in terms of achieving high detection accuracy compared to shallow tuning and low computational complexity compared to deep tuning. Therefore, all the hyperparameters of the employed CNN algorithms have optimized and fine-tuned in the proposed AMD model until an efficient and high detection rate is achieved. After running many tests and experiments, the final fine-tuning and optimization parameters used in the proposed vision-based AMD model are: learning rate of 0.00001, ADAM optimizer [52], ridge regression regularizer (L2-regularization) [53] with a weight decay rate of 0.001, maximum number of epochs equals 10, minimum batch size of 16, validation frequency of 16, a dropout rate of 0.5, learnRateSchedule parameter is set to be ''piecewise'', LearnRateDropPeriod parameter is set to 3, LearnRateDropFactor parameter is set to 0.9, and loss categorical cross-entropy function is used. These all fine-tuned hyperparameters were carefully chosen to avoid the overfitting occurrence and optimize the performance of the training and validation processes.

Furthermore, in the whole employed CNN algorithms, the softmax and fully-connected classifiers were utilized to classify between Android malware and benign samples. Thus, the output layer in the employed 16 different CNN algorithms that includes 1000 classes is customized and fine-tuned to have only two classes (malware and benign). Also, the back-propagation technique [54] is utilized in the proposed AMD model to fine-tune and optimize the hyberparameters and weights of the layers in the employed CNN algorithms that were initially trained on the ImageNet dataset; this is to achieve high detection efficiency in identifying malware attacks.

C. DETECTION EVALUATION MODULE

The detection evaluation module is concerned with comprehensively evaluating the proposed vision-based AMD model using 16 different detection assessment parameters. Consequently, the classification and detection efficiency of the suggested 16 different CNN algorithms have examined in terms of (1) recognition accuracy, (2) recall (sensitivity) (TPR) (true positive rate), (3) precision (PPV) (positive predictive value), (4) NPV (negative predictive value), (5) specificity (TNR) (true negative rate), (6) FNR (false negative rate), (7) FPR (false positive rate), (8) FOR (false omission rate), (9) FDR (false discovery rate), (10) misclassification rate, (11) F1-Score, (12) AROC (Area under the receiver operating characteristic) score, (13) accuracy curve, (14) loss curve, (15) confusion matrix, and (16) AROC curve. Further details and explanations of these detection assessment parameters can be explored in [55], [56], and they can be mathematically expressed as follows:

$$\text{Accuracy} = \frac{TN + TP}{FP + TP + FN + TN} \tag{1}$$

$$\text{Sensitivity} = \text{Recall (TPR)} = \frac{TP}{FN + TP} \tag{2}$$

$$\text{Precision (PPV)} = \frac{TP}{FP + TP} \tag{3}$$

$$\text{NPV} = \frac{TN}{FN + TN} \tag{4}$$

$$\text{Specificity (TNR)} = \frac{TN}{FP + TN} \tag{5}$$

$$\text{FNR} = \frac{FN}{TP + FN} \tag{6}$$

$$\text{FPR} = \frac{FP}{TN + FP} \tag{7}$$

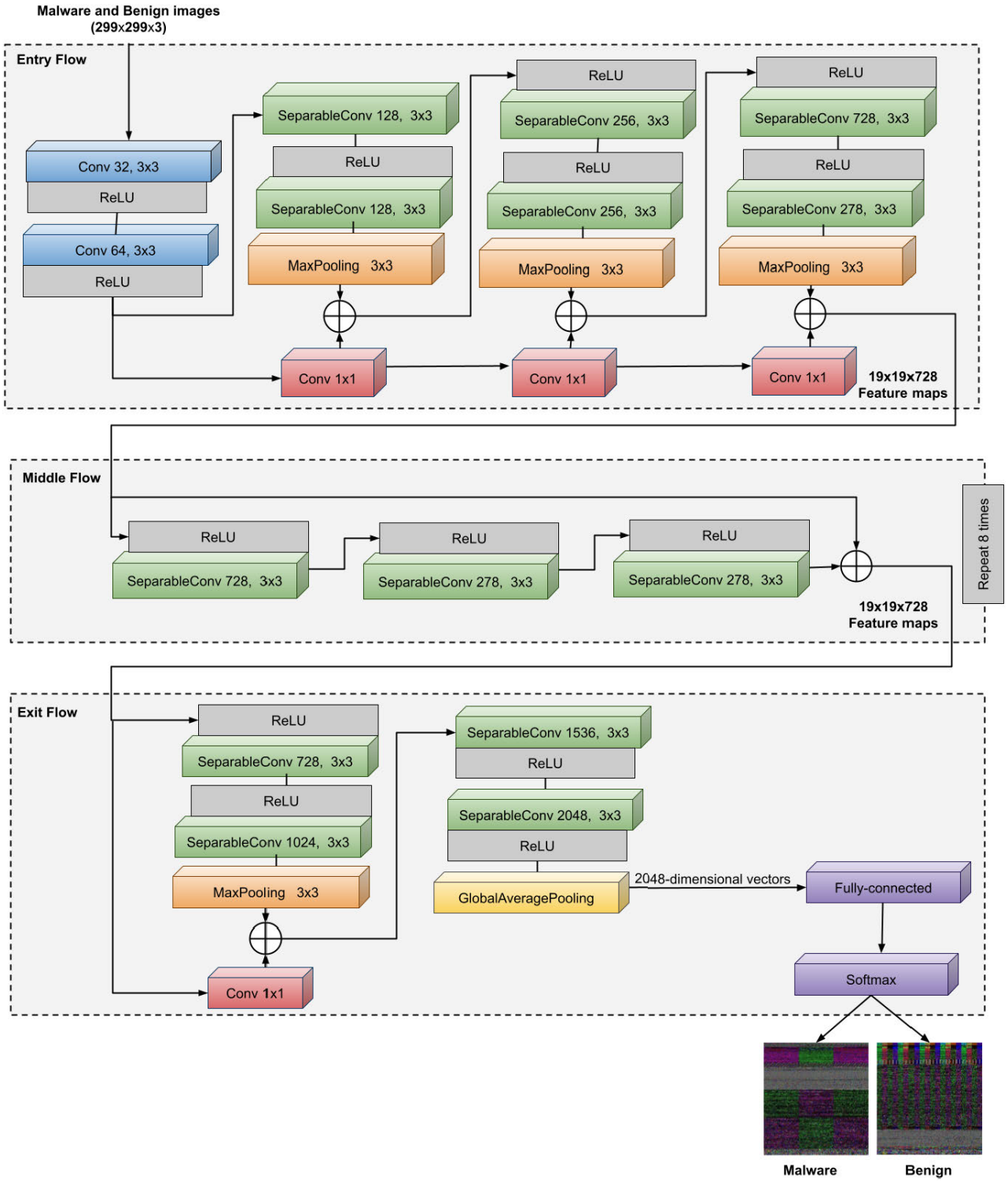


FIGURE 4. Flow structure of the fine-tuned Xception CNN algorithm.

$$\text{FOR} = \frac{FN}{TN + FN} \tag{8}$$

$$\text{FDR} = \frac{FP}{TP + FP} \tag{9}$$

$$\text{Misclassification rate} = \frac{FN + FP}{FP + TP + FN + TN} \tag{10}$$

$$\text{F1-Score} = \frac{2TP}{2TP + FN + FP} \tag{11}$$

where *TN* (true negative), *TP* (true positive), *FN* (false negative), and *FP* (false positive) are estimated by the confusion matrix shown in Fig. 5. The confusion matrix is called an error matrix, where it visualizes the different output predictions of the analyzed detection task. *TP* means that the prediction output is positive and it is actually positive, *FN* means that the prediction output is negative, but it is actually positive, *FP* means that the prediction output is positive, while it is actually negative, and *TN* means that the prediction output is negative and it is actually negative.

The ROC curve demonstrates the graphical representation of the tradeoff relationship between the TPR and FPR (1-specificity). ROC score is the average value of the area under the ROC curve. The accuracy curve is a graphical representation that reflects the tracing curve with the accuracy percentage for all training iterations (epochs). In contrast, the loss curve is a graphical representation that reflects the tracing curve with the loss percentage for all training iterations (epochs).

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

This section introduces extensive experimental results, more discussions, and comprehensive detection and complexity analysis for the performance validation of the proposed AMD model. Leopard Android dataset that originally contained an imbalanced number of malware and benign APKs was used by the proposed model. All training and classification simulations are performed using MATLAB 2020b software on a personal laptop with 8GB RAM and Intel Core i7-4500 processor.

In the experimental results, two binary classification scenarios for validating the performance of the proposed vision-based AMD model are tested. The first scenario tested the detection efficiency of the proposed model using imbalanced Android samples, as presented in section IV-A. The second scenario tested the detection efficiency of the proposed model using balanced Android samples, as introduced in section IV-B. These two classification scenarios are investigated to demonstrate the succeeding performance of the developed fine-tuned CNN algorithms even when applied on different sizes of balanced and imbalanced datasets without utilizing data augmentation techniques employed in the conventional malware detection and classification approaches.

Furthermore, for these two aforementioned classification scenarios, additional extensive experiments for the developed 16 fine-tuned CNN algorithms utilized in the proposed AMD model have been presented using two different image modalities. So, the detection accomplishment of the suggested vision-based AMD model was validated using both visual color and grayscale images of Android benign and malware apps. These comprehensive experiments have been run for the suggested AMD model to confirm its high detection capability and elevated classification efficiency on different representations of visual images.

For simplicity in displaying the detection evaluation outcomes, the confusion matrix, the loss & accuracy curves, the

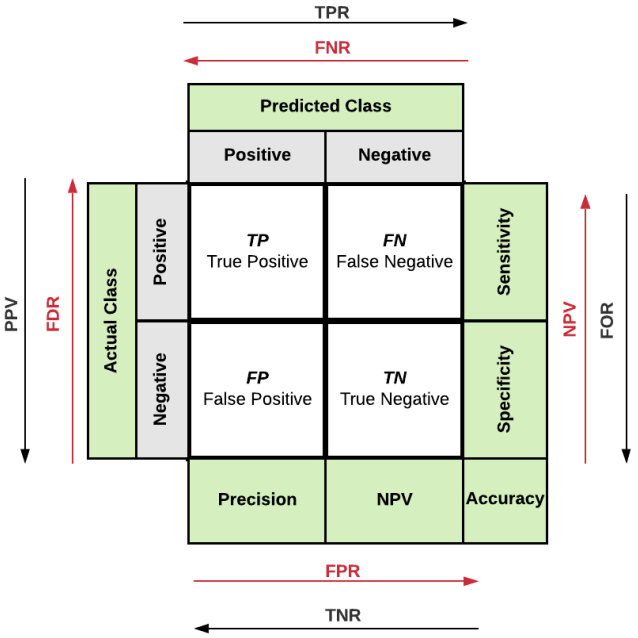


FIGURE 5. Binary confusion matrix.

ROC curve, and the other estimated assessment parameters are introduced in detail for Xception, the superior performed fine-tuned CNN algorithm amongst the 16 different tested CNN algorithms. In addition, the average outcomes of all estimated detection assessment metrics have been offered for the other CNN algorithms to deliver in-depth comparisons and evaluations among them.

In addition, a complexity analysis in terms of the storage capacity and execution time of the utilized Android datasets and CNN algorithms is presented in section IV-C. Finally, a comparative detection analysis between the proposed vision-based AMD model and other recent AMD models that used the same Android Leopard mobile apps dataset is discussed in section IV-D. This comparative study is presented to confirm the superiority of the proposed automated vision-based AMD model in comparison of recent related and conventional DL and ML-based AMD models in detecting and identifying Android malware attacks.

A. PERFORMANCE ANALYSIS ON COLOR AND GRAYSCALE IMAGES OF IMBALANCED ANDROID SAMPLES

This section provides the performance analysis of the proposed vision-based AMD model using imbalanced color and grayscale visual images (14733 malware images and 2486 benign images). So, more experiments were carried out for testing the proposed model performance using the 16 different fine-tuned CNN algorithms utilizing the imbalanced color and grayscale malware and benign images.

The training and testing accuracy & loss curves of the superior fine-tuned Xception CNN algorithm utilizing the imbalanced visual color and grayscale images across ten epochs are demonstrated in Figs. 6 and 7, respectively. It is observed from these curves that both the accuracy and loss

curves of the training and testing processes for the color and grayscale images are compatible with each other. There is only little overfitting in the loss curves, resulting from the imbalanced samples of both visual color and grayscale images of the malware and benign apps. But the validation losses in both image cases are still lower than 0.1 at epoch 10, which are acceptable values. In general, both curves for the loss and accuracy of the training and testing operations of the imbalanced visual color and grayscale images were stable before less than five epochs. Thus, as noticed, the employed Xception CNN algorithm achieved high detection efficiency at a lower number of iterations (epochs) for both imbalanced color and grayscale images. So, it is highly advocated for recognizing malware attacks efficiently and accurately in Android cybersecurity applications. Similarly, it is noticed analogous loss and accuracy curves for the other examined 15 different fine-tuned CNN algorithms of all tested experimental scenarios on color and grayscale images.

The confusion matrices obtained for the superior fine-tuned Xception CNN algorithm utilizing the imbalanced visual color and grayscale images are presented in Fig. 8. These are binary confusion matrices for the examined benign and malware color and grayscale images of the imbalanced Android samples. It is observed that the accomplished *TP, FP, TN*, and *FN* values for the visual color images are better than those of the grayscale images. But, in general, the obtained values of both image visualizations for the fine-tuned Xception CNN algorithm were acceptable, especially in the detection situation of the highly imbalanced Android datasets. Thus, the fine-tuned Xception CNN algorithm accomplished 98.05% and 97.93% of accuracy in correctly detecting malware and benign samples for imbalanced color images and imbalanced grayscale images, respectively. These results are also confirmed and supported by the obtained outcomes for the fine-tuned Xception CNN algorithm that achieved high sensitivity, specificity, and ROC values of 0.9095, 0.9925, and 0.9957, respectively, for the visual color images. Also, this CNN algorithm attained high sensitivity, specificity, and ROC values of 0.9074, 0.9915, and 0.9953, respectively, for the visual grayscale images. These all achieved results are excellent due to exploiting the benefits of transfer learning and fine-tuning the hyperparameters and CNN layers of the suggested CNN algorithms.

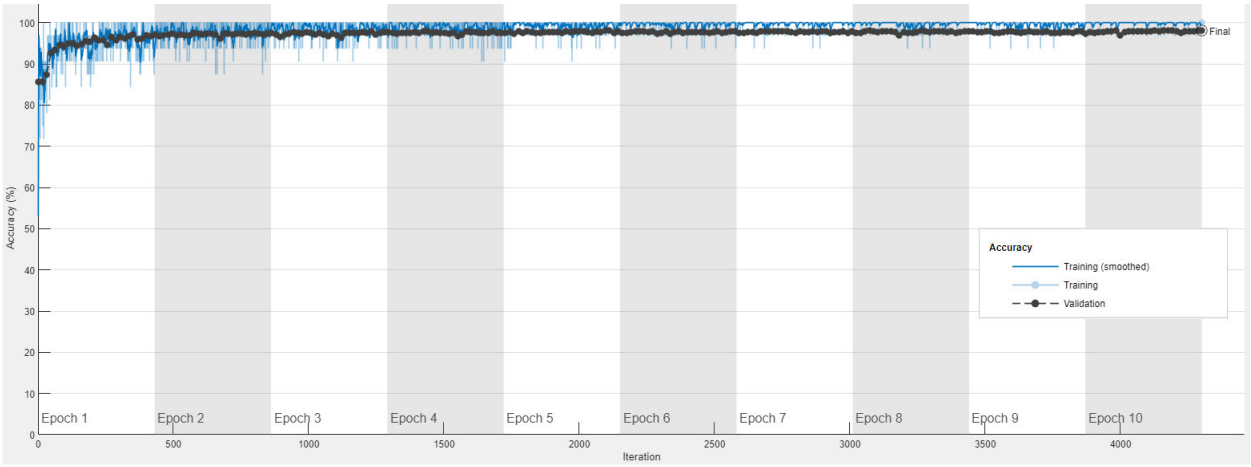
In addition, the detection performance capability of all 16 analyzed fine-tuned CNN algorithms in recognizing color or grayscale benign and malware images have quantitatively examined. So, the accuracy (Acc.), recall (Rec.), precision (Prec.), NPV, specificity (Spec.), FNR, FPR, FOR, FDR, misclassification rate (Mis. Class. Rate), F1-Score, and AROC score are computed for the suggested CNN algorithms. Table 4 demonstrates the detection outcomes of the employed CNN algorithms on the imbalanced visual color images. Similarly, the detection outcomes of the examined CNN algorithms on the imbalanced visual grayscale images are depicted in Table 5. These obtained detection comparisons disclosed that the proposed fine-tuned Xception CNN algorithm accomplishes superior and substantial values than the other CNN algorithms for all considered and calculated detection assessment parameters for both color and grayscale image representations. Consequently, this CNN algorithm is remarkably advised to detect Android malware attacks of visualized Android apps effectively.

B. PERFORMANCE ANALYSIS ON COLOR AND GRAYSCALE IMAGES OF BALANCED ANDROID SAMPLES

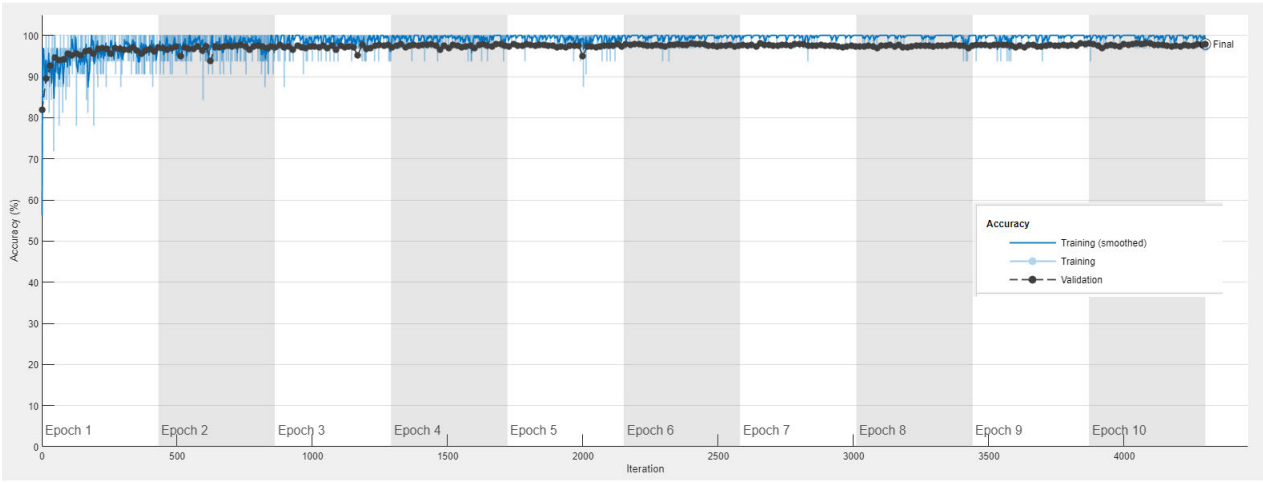
This section provides the performance analysis of the proposed vision-based AMD model using balanced color and grayscale visual images (2486 malware images and 2486 benign images). So, more experiments were carried out for testing the proposed model performance using the 16 different fine-tuned CNN algorithms utilizing the balanced color and grayscale malware and benign images.

The training and testing accuracy & loss curves of the superior fine-tuned Xception CNN algorithm utilizing the balanced visual color and grayscale images across ten epochs are demonstrated in Figs. 9 and 10, respectively. It is observed from these curves that both the accuracy and loss curves of the training and testing processes for the color and grayscale images are fully compatible with each other. In general, both curves for the loss and accuracy of the training and testing operations of the balanced visual color and grayscale images were stable before less than five epochs. Thus, as noticed, the employed Xception CNN algorithm achieved high detection efficiency at a lower number of iterations (epochs) for both balanced color and grayscale images. So, it is highly advocated for recognizing malware attacks efficiently and accurately in Android cybersecurity applications. Similarly, it is noticed analogous loss and accuracy curves for the other examined 15 different fine-tuned CNN algorithms of all tested experimental scenarios on color and grayscale images.

The confusion matrices obtained for the superior fine-tuned Xception CNN algorithm utilizing the balanced visual color and grayscale images are presented in Fig. 11. These are binary confusion matrices for the examined benign and malware color and grayscale images of the balanced Android samples. It is observed that the accomplished *TP, FP, TN*, and *FN* values for the visual color images are better than those of the grayscale images. But, in general, the obtained values of both image visualizations for the fine-tuned Xception CNN algorithm were highly recommended and good. These all attained results are excellent due to exploiting the benefits of transfer learning and fine-tuning the hyperparameters and CNN layers of the suggested CNN algorithms. Thus, the fine-tuned Xception CNN algorithm achieved malware and benign samples detection accuracy that reached 99.40% and 99.20% for balanced color images and balanced grayscale images, respectively. These results are also confirmed and supported by the obtained outcomes for the fine-tuned Xception CNN algorithm that achieved high sensitivity, specificity, and ROC values of 0.9940, 0.9940, and 0.9995, respectively, for the visual color images. Also,



(a) For visual color images.

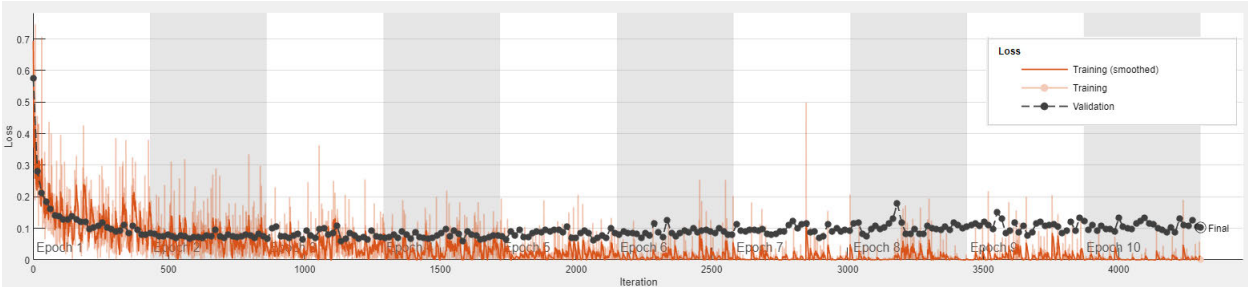


(b) For visual grayscale images.

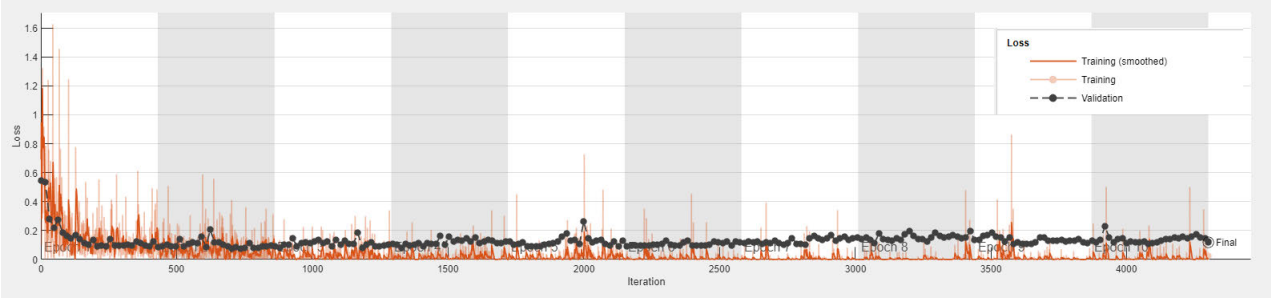
FIGURE 6. Training and testing accuracy curves of the superior fine-tuned Xception CNN algorithm on imbalanced samples of (a) visual color images and (b) visual grayscale images.

TABLE 4. Outcomes of detection assessment of the employed CNN algorithms on the imbalanced visual color images.

CNN algorithm	Acc.	Rec. (TPR)	Prec. (PPV)	NPV	Spec. (TNR)	FNR	FPR	FOR	FDR	Mis Class. Rate	F1-Score	AROC
Xception	0.9805	0.9095	0.9535	0.9848	0.9925	0.0905	0.0075	0.0152	0.0464	0.0194	0.931	0.9957
VGG16	0.9803	0.9316	0.9316	0.9885	0.9885	0.0684	0.0115	0.0115	0.0684	0.0197	0.9316	0.9872
VGG19	0.9774	0.8994	0.9411	0.9832	0.9905	0.1006	0.0095	0.0168	0.0589	0.0226	0.9198	0.9934
DarkNet53	0.9765	0.9276	0.9111	0.9877	0.9847	0.0724	0.0153	0.0123	0.0889	0.0235	0.9192	0.9937
MobileNetV2	0.9765	0.9175	0.9194	0.9861	0.9864	0.0824	0.0136	0.0139	0.0806	0.0235	0.9184	0.9906
ResNet101	0.9765	0.9095	0.9262	0.9848	0.9877	0.0905	0.0122	0.0152	0.0738	0.0235	0.9178	0.9795
AlexNet	0.9762	0.9074	0.9261	0.9844	0.9878	0.0925	0.0122	0.0156	0.0739	0.0238	0.9167	0.9937
ResNet50	0.9747	0.9115	0.9133	0.9851	0.9854	0.0885	0.0146	0.0149	0.0867	0.0253	0.9124	0.9779
ResNet18	0.9747	0.8893	0.9325	0.9815	0.9891	0.1107	0.0109	0.0185	0.0675	0.0253	0.9104	0.9879
InceptionV3	0.9747	0.8853	0.9362	0.9808	0.9898	0.1147	0.0102	0.0192	0.0638	0.0253	0.91	0.983
DarkNet19	0.9739	0.8934	0.9231	0.9821	0.9874	0.1066	0.0126	0.0179	0.0769	0.0261	0.908	0.9934
ShuffleNet	0.9721	0.8954	0.91	0.9824	0.9851	0.1046	0.0149	0.0176	0.0899	0.0279	0.9026	0.9926
Places365-GoogleNet	0.9704	0.9276	0.8748	0.9877	0.9776	0.0724	0.0224	0.0123	0.1252	0.0296	0.9004	0.9941
NasNetMobile	0.9701	0.8712	0.9174	0.9785	0.9867	0.1288	0.0132	0.0215	0.0826	0.0299	0.8937	0.9924
GoogleNet	0.9668	0.9376	0.8488	0.9893	0.9718	0.0624	0.0282	0.0107	0.1512	0.0331	0.891	0.9907
SqueezeNet	0.9663	0.8551	0.9062	0.9758	0.985	0.1449	0.0149	0.0242	0.0938	0.0337	0.8799	0.9911

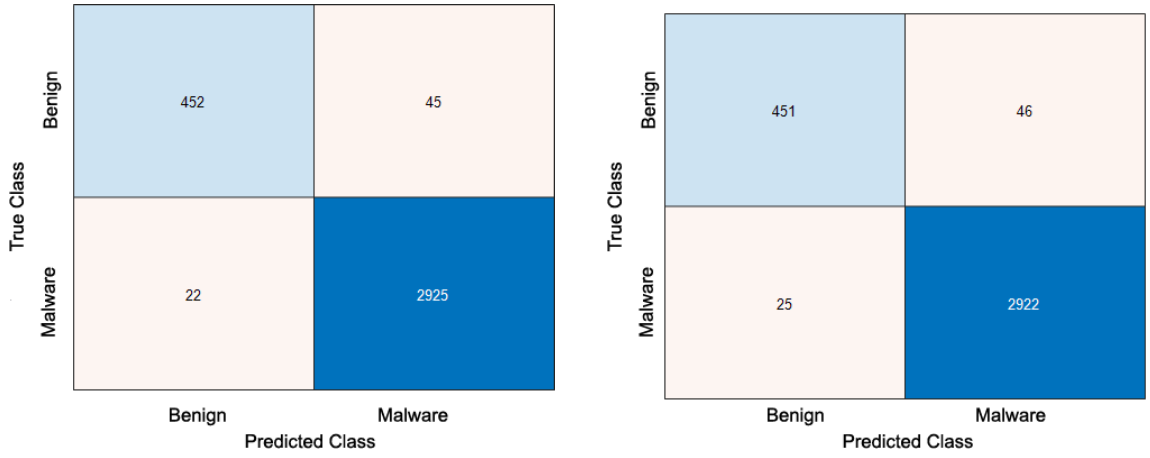


(a) For visual color images.



(b) For visual grayscale images.

FIGURE 7. **Training and testing loss curves of the superior fine-tuned Xception CNN algorithm on imbalanced samples of (a) visual color images and (b) visual grayscale images.**



(a) For visual color images.

(b) For visual grayscale images.

FIGURE 8. **Confusion matrix of the superior fine-tuned Xception CNN algorithm on imbalanced samples of (a) visual color images and (b) visual grayscale images.**

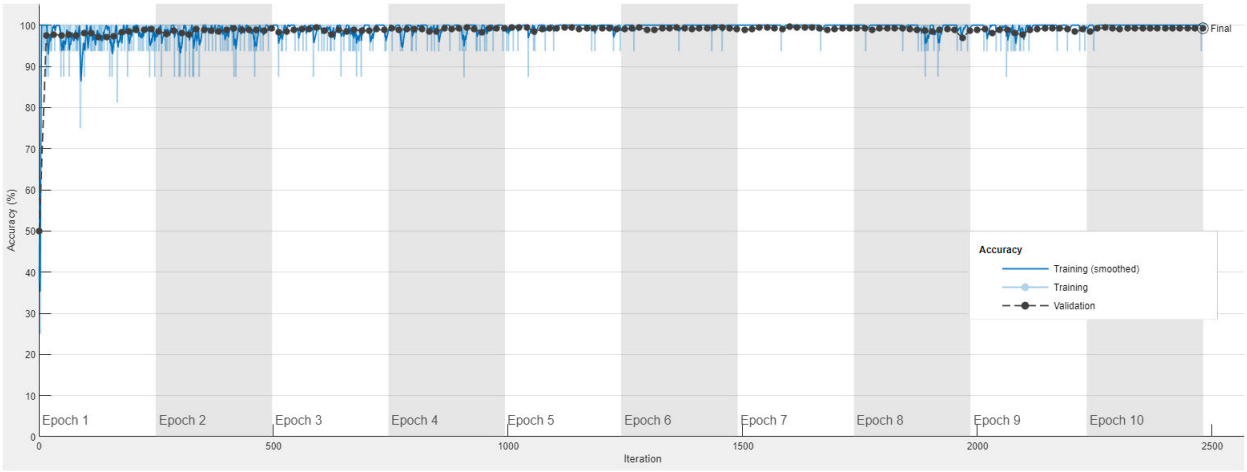
this CNN algorithm attained high sensitivity, specificity, and ROC values of 0.9920, 0.9920, and 0.9998, respectively, for the visual grayscale images.

Furthermore, the detection performance capability of all 16 analyzed fine-tuned CNN algorithms in recognizing balanced color or grayscale benign and malware images has examined. So, the accuracy (Acc.), recall (Rec.), precision (Prec.), NPV, specificity (Spec.), FNR, FPR, FOR, FDR, misclassification rate (Mis. Class. Rate), F1-Score, and AROC are computed for the suggested CNN algorithms. Table 6 demonstrates the detection outcomes of the employed CNN algorithms on the balanced visual color images, while the detection outcomes of the empoyed CNN algorithms on the imbalanced visual grayscale images as depicted in Table 7. These obtained detection comparisons disclosed that the proposed fine-tuned Xception CNN algorithm accomplishes superior and substantial values than the other CNN algorithms for all considered and calculated detection assessment parameters for balanced color and grayscale image representations. Consequently, this CNN algorithm is remarkably advised to detect Android malware attacks of visualized Android apps effectively.

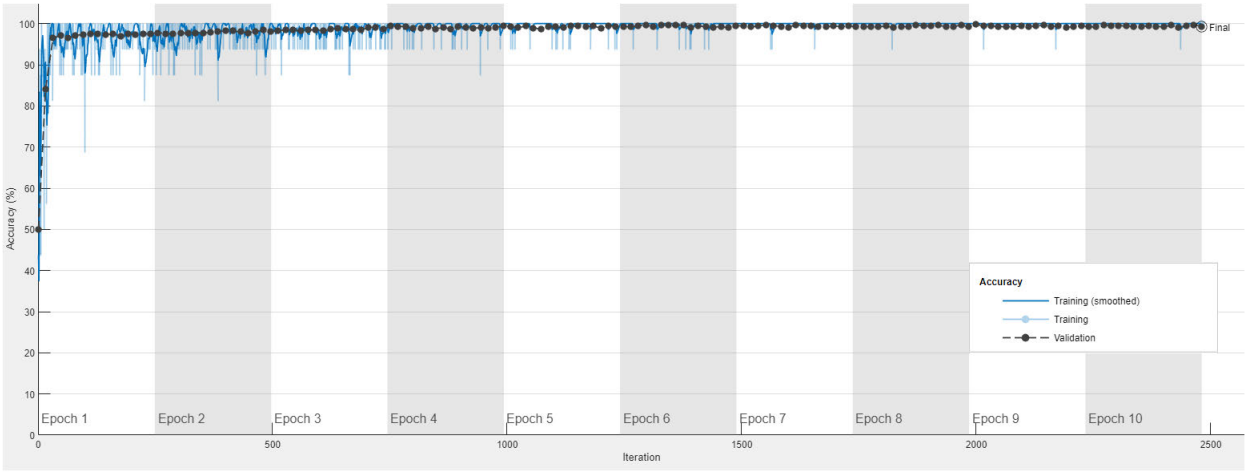
Overall, the whole examined and suggested fine-tuned CNN algorithms accomplished recommended detection findings, and thus, they can be utilized effectively for detecting

TABLE 5. Outcomes of detection assessment of the best four performed CNN algorithms on the imbalanced visual grayscale images.

CNN algorithm	Acc.	Rec. (TPR)	Prec. (PPV)	NPV	Spec. (TNR)	FNR	FPR	FOR	FDR	Mis Class. Rate	F1-Score	AROC
Xception	0.9793	0.9074	0.9475	0.9845	0.9915	0.0926	0.0084	0.0155	0.0525	0.0206	0.927	0.9953
VGG16	0.9785	0.8994	0.949	0.9832	0.9919	0.1006	0.0081	0.0168	0.0509	0.0215	0.9235	0.9948
VGG19	0.9762	0.9296	0.9077	0.9881	0.9841	0.0704	0.0159	0.0119	0.0923	0.0238	0.9185	0.992
ResNet101	0.9762	0.8994	0.9332	0.9831	0.9891	0.1006	0.0108	0.0169	0.0668	0.0238	0.9159	0.9883
DarkNet53	0.9759	0.9155	0.9173	0.9858	0.9861	0.0845	0.0139	0.0142	0.0826	0.0241	0.9164	0.9947
AlexNet	0.9756	0.8913	0.9366	0.9818	0.9898	0.1087	0.0102	0.0182	0.0634	0.0244	0.9134	0.9852
InceptionV3	0.9747	0.9215	0.9051	0.9867	0.9837	0.0785	0.0163	0.0133	0.0949	0.0253	0.9133	0.9929
ResNet18	0.9744	0.8812	0.9379	0.9802	0.9902	0.1187	0.0098	0.0198	0.0621	0.0256	0.9087	0.983
ResNet50	0.9721	0.8954	0.91	0.9824	0.9851	0.1046	0.0149	0.0176	0.0899	0.0279	0.9026	0.9926
DarkNet19	0.9704	0.9276	0.8748	0.9877	0.9776	0.0724	0.0224	0.0123	0.1252	0.0296	0.9004	0.9941
Places365-GoogleNet	0.9701	0.9417	0.8635	0.99	0.9749	0.0584	0.0251	0.0099	0.1365	0.0299	0.9008	0.9934
GoogleNet	0.9698	0.8632	0.9226	0.9772	0.9878	0.1368	0.0122	0.0228	0.0774	0.0302	0.8919	0.9919
MobileNetV2	0.9689	0.8833	0.8996	0.9804	0.9834	0.1167	0.0166	0.0196	0.1004	0.0311	0.8914	0.9907
ShuffleNet	0.9689	0.8853	0.898	0.9807	0.983	0.1146	0.017	0.0193	0.102	0.0311	0.8916	0.9899
NasNetMobile	0.9663	0.8551	0.9062	0.9758	0.985	0.1449	0.0149	0.0242	0.0938	0.0337	0.8799	0.9911
SqueezeNet	0.9512	0.9155	0.7831	0.9853	0.9572	0.0845	0.0428	0.0147	0.2169	0.0488	0.8442	0.9711



(a) For visual color images.



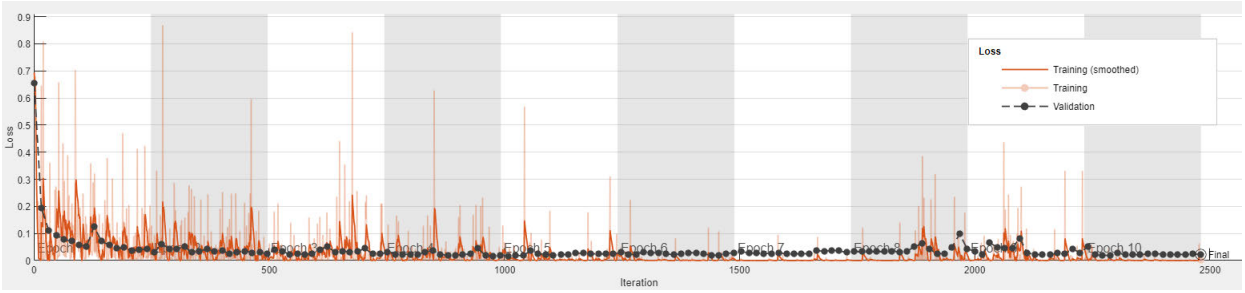
(b) For visual grayscale images.

FIGURE 9. Training and testing accuracy curves of the superior fine-tuned Xception CNN algorithm on balanced samples of (a) visual color images and (b) visual grayscale images.

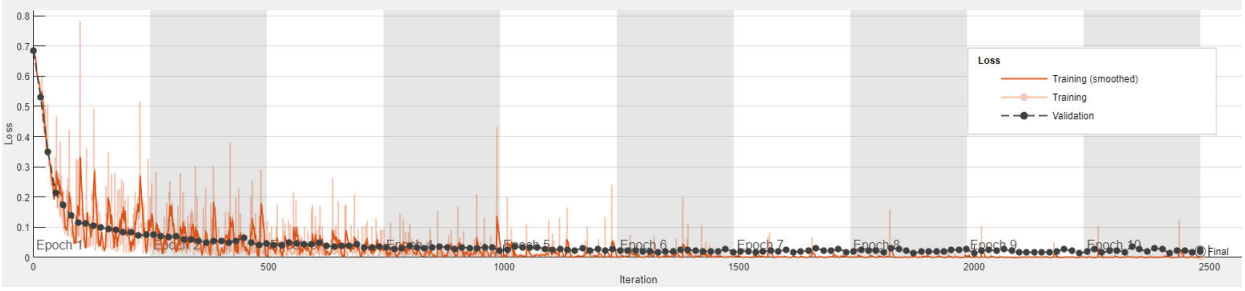
malware attacks in the form of visual color or grayscale images using imbalanced or balanced Android datasets. Furthermore, as observed, the whole accomplished results for the visual color images are better than those of accomplished

TABLE 6. Outcomes of detection assessment of the employed CNN algorithms on the balanced visual color images.

CNN algorithm	Acc.	Rec. (TPR)	Prec. (PPV)	NPV	Spec. (TNR)	FNR	FPR	FOR	FDR	Mis Class. Rate	F1-Score	AROC
Xception	0.994	0.994	0.9939	0.994	0.994	0.006	0.006	0.006	0.006	0.006	0.994	0.9995
VGG16	0.993	0.9899	0.9959	0.99	0.996	0.0101	0.004	0.01	0.004	0.007	0.9929	0.9998
VGG19	0.992	0.996	0.988	0.9959	0.9879	0.004	0.0121	0.0041	0.012	0.0081	0.992	0.9979
DarkNet53	0.9909	0.992	0.9899	0.9919	0.9899	0.008	0.0101	0.008	0.01	0.0091	0.9909	0.9992
DarkNet19	0.9899	0.998	0.9822	0.998	0.9819	0.002	0.0181	0.002	0.0178	0.0101	0.99	0.9973
AlexNet	0.9899	0.996	0.9841	0.9959	0.9839	0.004	0.0161	0.0041	0.0159	0.0101	0.99	0.9963
ResNet18	0.9899	0.992	0.9879	0.9919	0.9879	0.008	0.0121	0.0081	0.012	0.0101	0.9899	0.9997
ResNet101	0.9889	0.9899	0.988	0.9899	0.9879	0.0101	0.0121	0.0101	0.012	0.0111	0.9889	0.9961
NasNetMobile	0.9879	0.9859	0.9899	0.9859	0.9899	0.0141	0.01	0.014	0.0101	0.0121	0.9879	0.9994
GoogleNet	0.9879	1	0.9764	1	0.9759	0	0.0241	0	0.0236	0.0121	0.9881	0.9983
InceptionV3	0.9869	0.9879	0.9859	0.9879	0.9859	0.0121	0.0141	0.0121	0.0141	0.0131	0.9869	0.9974
MobileNetV2	0.9859	0.9859	0.9859	0.9859	0.9859	0.0141	0.0141	0.0141	0.0141	0.0141	0.9859	0.9995
Places365-GoogleNet	0.9859	0.9839	0.9879	0.984	0.9879	0.0161	0.0121	0.016	0.0121	0.0141	0.9859	0.999
ShuffleNet	0.9859	0.992	0.9801	0.9919	0.9799	0.0081	0.0201	0.0081	0.0199	0.0141	0.986	0.9957
ResNet50	0.9839	0.9899	0.9781	0.9898	0.9779	0.0101	0.0221	0.0102	0.0218	0.0161	0.984	0.9908
SqueezeNet	0.9818	0.9899	0.9743	0.9898	0.9738	0.0101	0.0262	0.0102	0.0257	0.0181	0.982	0.9924



(a) For visual color images.



(b) For visual grayscale images.

FIGURE 10. Training and testing loss curves of the superior fine-tuned Xception CNN algorithm on balanced samples of (a) visual color images and (b) visual grayscale images.

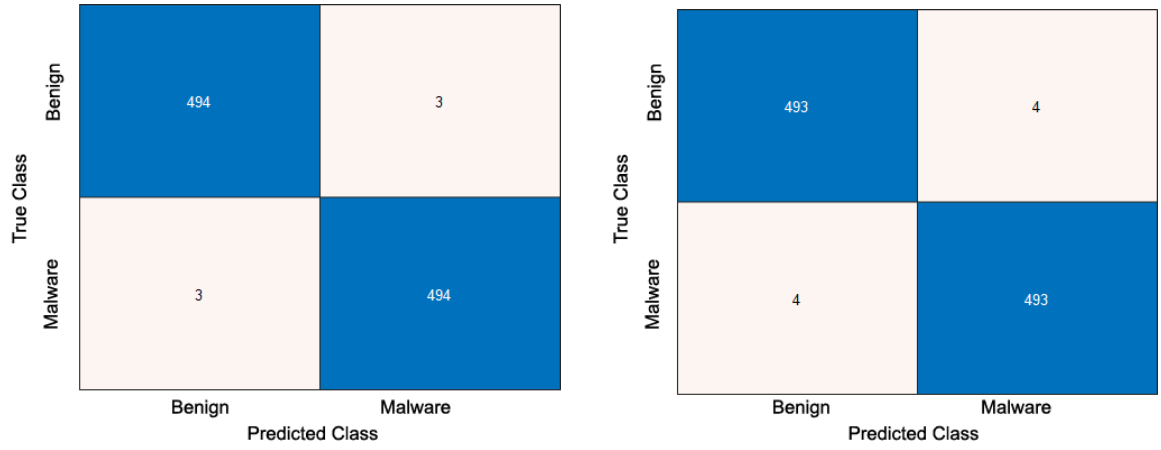
results for the visual grayscale images either for balanced or imbalanced samples. This is because color images contain more visualization features and texture details than those included in the grayscale images. Also, as noticed, the whole obtained detection results on testing the balanced Android samples are better than those obtained for testing imbalanced Android samples for all examined fine-tuned CNN algorithms.

C. COMPLEXITY ANALYSIS

This section discusses the complexity performance in terms of the storage capacity, experimental analysis, and execution time of the utilized Android datasets and CNN algorithms. So, the quantitative computational analysis of the utilized CNN algorithms in the proposed automated vision-based AMD model is examined in terms of (1) storage capacity of the used color and grayscale Android samples of the imbalanced and balanced datasets, (2) experimental analysis in terms of the (a) number of layers, (b) storage capacity, (c) total number of the trainable and non-trainable parameters, and (d) reduction percentage in the training parameters of the examined CNN algorithms used in the detection experiments, and (3) execution time analysis of the examined CNN algorithms of the color and grayscale Android samples of the imbalanced and balanced datasets.

TABLE 7. Outcomes of detection assessment of the employed CNN algorithms on the balanced visual gray-scale images.

CNN algorithm	Acc.	Rec. (TPR)	Prec. (PPV)	NPV	Spec. (TNR)	FNR	FPR	FOR	FDR	Mis Class. Rate	F1-Score	AROC
Xception	0.992	0.992	0.992	0.992	0.992	0.0081	0.0081	0.0081	0.0081	0.0081	0.992	0.9998
VGG16	0.992	0.9899	0.994	0.9899	0.994	0.0101	0.006	0.01	0.0061	0.008	0.9919	0.9997
VGG19	0.9899	0.9839	0.9959	0.9841	0.996	0.0161	0.004	0.0159	0.0041	0.01	0.9899	0.9994
DarkNet53	0.9859	0.9839	0.9879	0.984	0.9879	0.0161	0.0121	0.016	0.0121	0.0141	0.9859	0.9974
DarkNet19	0.9859	0.9839	0.9879	0.9839	0.9879	0.0161	0.0121	0.016	0.0121	0.0141	0.9859	0.9966
AlexNet	0.9859	0.9879	0.984	0.9879	0.9839	0.0121	0.0161	0.0121	0.016	0.0141	0.9859	0.9972
ResNet18	0.9859	0.9859	0.9859	0.9859	0.9859	0.0141	0.0141	0.0141	0.0141	0.0141	0.9859	0.9979
MobileNetV2	0.9859	0.9899	0.982	0.9899	0.9819	0.0101	0.0181	0.0101	0.018	0.0141	0.986	0.9973
Places365-GoogleNet	0.9859	0.9839	0.9879	0.984	0.9879	0.0161	0.0121	0.016	0.0121	0.0141	0.9859	0.9974
SqueezeNet	0.9849	0.998	0.9725	0.9979	0.9718	0.002	0.0282	0.0021	0.0275	0.0151	0.9851	0.996
NasNetMobile	0.9839	0.9859	0.982	0.9859	0.9818	0.0141	0.0181	0.0141	0.018	0.0161	0.9839	0.9988
GoogleNet	0.9819	0.9819	0.9819	0.9819	0.9819	0.0181	0.0181	0.0181	0.0181	0.0181	0.9819	0.9984
InceptionV3	0.9809	0.9818	0.9799	0.9818	0.9798	0.0181	0.0201	0.0181	0.0201	0.0191	0.9809	0.9973
ShuffleNet	0.9809	0.9899	0.9723	0.9898	0.9718	0.0101	0.0282	0.0102	0.0277	0.0191	0.9811	0.9985
ResNet101	0.9798	0.996	0.9649	0.9958	0.9638	0.004	0.0362	0.0042	0.0351	0.0201	0.9802	0.9952
ResNet50	0.9759	0.9738	0.9778	0.9739	0.9779	0.0262	0.0221	0.0261	0.0222	0.0241	0.9758	0.9942



(a) For visual color images.

(b) For visual grayscale images.

FIGURE 11. Confusion matrix of the superior fine-tuned Xception CNN algorithm on balanced samples of (a) visual color images and (b) visual grayscale images.

TABLE 8. Total number of Android malware and benign samples over imbalanced and balanced datasets.

	Imbalanced Samples	Balanced Samples
Malware	14733	2486
Benign	2486	2486
Total	17219	4972

Table 8 shows the total number of Android malware and benign samples over imbalanced and balanced datasets used in the experiments. Table 9 introduces storage capacity analysis of the color and grayscale Android samples of the imbalanced and balanced datasets. It is noticed from the last row in Table 9 that color images cause an extra storage capacity of 3.4% compared to grayscale images for the imbalanced Android samples and 5.3% for the balanced Android samples. However, this additional storage capacity is considered low comparing to the achieved detection accuracy when using color images, as explained in subsections IV-A and IV-B.

TABLE 9. Storage capacity (MB) analysis of the color and grayscale Android samples of the imbalanced and balanced datasets.

	Imbalanced Samples		Balanced Samples	
	Color	Gray	Color	Gray
Malware	526	506	67.1	59.2
Benign	120	118	120	118
Total	646	624	187.1	177.2
Extra storage caused by color images	3.4%		5.3%	

Table 10 presents the experimental analysis of the examined CNN algorithms used in the detection experiments. This experimental analysis is represented by (i) size (storage capacity) of the employed CNN algorithm on a disk, (ii) depth (layers) of the studied CNN algorithm containing the number of successive parallel or series fully connected or convolutional layers on a path from the input layer to the output layer utilized for feature extraction purposes, (iii) total

TABLE 10. Experimental analysis of the examined CNN algorithms used in the detection experiments.

CNN algorithm	Storage capacity (MB)	Depth (layers)	Total parameters	Trainable parameters	Non-trainable parameters	Reduced training parameters (%)
Xception	85	71	22,900,000	4,096	22,895,904	99.98
VGG16	515	16	138,000,000	8,192	137,991,808	99.99
VGG19	535	19	144,000,000	8,192	143,991,808	99.99
DarkNet53	155	53	41,600,000	2,048	41,597,952	99.99
MobileNetV2	13	53	3,500,000	2,560	3,497,440	99.93
ResNet101	167	101	44,600,000	4,096	44,595,904	99.99
AlexNet	227	8	61,000,000	8,192	60,991,808	99.99
ResNet50	96	50	25,600,000	4,096	25,595,904	99.98
ResNet18	44	18	11,700,000	1,024	11,698,976	99.99
InceptionV3	89	48	23,900,000	4,096	23,895,904	99.98
DarkNet19	78	19	20,800,000	2,048	20,797,952	99.99
ShuffleNet	5.4	50	7,000,000	1,024	6,998,976	99.99
Places365-GoogleNet	27	22	61,000,000	2,048	60,997,952	99.99
NasNetMobile	20	N/A	5,300,000	2,112	5,297,888	99.96
GoogleNet	27	22	7,000,000	2,048	6,997,952	99.97
SqueezeNet	5.2	18	1,240,000	1,024	1,238,976	99.92

TABLE 11. Execution time analysis of the examined CNN algorithms of the color and grayscale Android samples of the imbalanced and balanced datasets.

CNN algorithm	Execution time (sec)							
	Imbalanced				Balanced			
	Color		Gray		Color		Gray	
	Total time	Avg. time /APK	Total time	Avg. time /APK	Total time	Avg. time /APK	Total time	Avg. time /APK
Xception	8280	0.4809	7440	0.4321	3360	0.6758	2640	0.5309
VGG16	17040	0.9896	14160	0.8224	5820	1.1706	3720	0.7482
VGG19	14700	0.8537	8820	0.5122	5280	1.062	3480	0.6999
DarkNet53	11400	0.6621	9180	0.5331	3120	0.6275	2700	0.543
MobileNetV2	6120	0.3554	5340	0.3101	2220	0.4465	1920	0.3862
ResNet101	13080	0.7596	10320	0.5993	4140	0.8327	3780	0.7603
AlexNet	4500	0.2613	3780	0.2195	1080	0.2172	780	0.1569
ResNet50	9060	0.5262	7680	0.446	2220	0.4465	1920	0.3862
ResNet18	5940	0.3449	4860	0.2823	1320	0.2655	1020	0.2052
InceptionV3	12300	0.7143	11340	0.6586	4320	0.8689	3780	0.7603
DarkNet19	7620	0.4425	6120	0.3554	1560	0.3138	1230	0.2474
ShuffleNet	6960	0.4042	5880	0.3415	2070	0.4163	1800	0.362
Places365-GoogleNet	6300	0.3659	5520	0.3206	1590	0.3197	1260	0.2534
NasNetMobile	18960	1.1011	16080	0.9339	11880	2.3894	8520	1.7136
GoogleNet	6420	0.3728	5160	0.2997	1560	0.3138	1440	0.2896
SqueezeNet	5280	0.3066	4080	0.2367	1140	0.2293	900	0.181

parameters of the tested CNN algorithm from the input layer to the output layer, (iv) trainable parameters of the unfrozen CNN layers, (v) non-trainable parameters of the frozen CNN layers, and (vi) reduced percentage in the training parameters of the examined CNN algorithm. It is observed from Table 10 that number of layers, non-trainable parameters, and trainable parameters vary from one CNN algorithm to another. Also, due to exploiting transfer learning advantages in the proposed AMD model, there is a considerable reduction in training parameters of the whole employed CNN algorithms. So, most of the layers and training parameters of the employed CNN algorithms were frozen, as discussed in subsection III-B. For example, the best accurate fine-tuned Xception algorithm used in the proposed vision-based AMD model trained only 4,096 parameters from 22,900,000 parameters that were existed in the original Xception CNN algorithm.

Table 11 illustrates the execution time analysis of the examined CNN algorithms for the color and grayscale Android samples of the imbalanced and balanced datasets. So, the computational overhead of the CNN algorithms used in the proposed vision-based AMD model is estimated in terms of (1) the total computational time of the validation and training processes and (2) the average computational time to identify Android malware or benign sample, which is calculated by dividing the whole computational time by the total number of Android malware and benign samples. It is noticed from Table 11 that the computational overhead is varied from one CNN algorithm to another due to the variation in the number of layers and parameters amongst the employed CNN algorithms as demonstrated in Table 10. Nevertheless, the obtained outcomes proved that the average computational time spent to detect Android malware or benign sample is adequate for all examined CNN algorithms. For example, the best accurate fine-tuned Xception algorithm used in the proposed vision-based AMD model accomplished low computational times of 0.4565 sec and 0.60335 sec to identify the Android sample in the imbalanced and balanced datasets, respectively.

TABLE 12. Comparison between the proposed vision-based AMD model and the recent conventional vision-based AMD models tested on the Leopard Android dataset.

Reference No.	Year	Acc. (%)	Technique	Dataset distribution	Visualization method	Augmentation or features-engineering
[8]	2018	93	CNN	Imbalanced	Color	No
[9]	2019	97.35	ML and CNN	Imbalanced	Color	Yes
[10]	2019	97.46	DCNN	Imbalanced	Color	No
[11]	2020	97.35	CNN	Imbalanced	Color	Yes
[28]	2020	97.81	DCNN	Imbalanced	Color	No
This work	-	**98.05**	**Fine-tuned CNN**	**Imbalanced**	**Color**	**No**
		97.93		**Imbalanced**	**Gray**	
		99.40		**Balanced**	**Color**	
		99.20		**Balanced**	**Gray**	

D. COMPARATIVE ANALYSIS WITH RELATED MODELS

This section compares the detection and classification performance of the proposed automated vision-based AMD model with the most recent vision-based AMD models. The purpose of this comparison is to highlight and demonstrate the proposed AMD model's superior accomplishment in recognizing and detecting visualized Android malware attacks using the Leopard mobile apps dataset, either with balanced or imbalanced Android samples.

Table 12 demonstrates the comparison between the proposed AMD model using fine-tuned Xception CNN algorithm and the recent conventional AMD models. It is remarked that the suggested AMD model achieved superior detection accuracy that reached 98.05% for imbalanced color dataset and 99.40% for balanced color dataset. These attained detection accuracies are higher than those of all other baseline-related AMD models that used the same Leopard Android mobile dataset. The proposed model did not employ any augmentation algorithms or feature engineering techniques like other conventional detection models.

Thus, in contrast to almost recent related vision-based AMD models that used some additional stages of feature-engineering or/and data augmentation techniques in their malware detection models, the proposed automated vision-based AMD model avoids the need for these computational stages. As observed, the proposed model achieved higher classification performance and higher detection efficiency than the conventional models by employing only transfer learning and fine-tuning algorithms for the utilized CNN algorithms; to detect Android malware attacks efficiently.

V. CONCLUSION AND FUTURE WORK

There are enormous limitations and difficulties in processing and analyzing unknown and massive Android malware samples using dynamic analysis, static analysis, or traditional ML techniques. Subsequently, there is an essential need to develop innovative artificial intelligence algorithms to mitigate the critical cybersecurity problems resulting from Android mobile malware attacks.

The vision-based DL techniques utilized for recognizing Android malware samples have significant detection merits by avoiding feature-engineering steps required to obtain hand-crafted features that increase the complexity of malware detection algorithms. Thus, this paper introduced an automated vision-based AMD model that composed 16 different fine-tuned CNN algorithms to efficiently and quickly detect Android malware attacks. The proposed AMD model was developed based on the visualization of Android APKs, transfer learning concept, and fine-tuning process; to proficiently classify benign APKs from malware APKs without extensive computations, reverse engineering, or feature extraction stages. Different experiments have been carried out using balanced and imbalanced Android samples of color and grayscale images generated from the benchmark Leopard dataset. The purpose of these comprehensive experiments is to extensively and sufficiently validate the detection and classification achievement of the suggested automated vision-based AMD model.

The experiments results of various classification assessment metrics revealed that the 16 different fine-tuned CNN algorithms included in the proposed AMD model have efficiently performed with the visualized color and grayscale images in case of balanced and imbalanced Android apps datasets. Moreover, compared to the related and conventional AMD models, the proposed AMD model achieved higher detection accuracy, lower computational overhead, and better recognition performance without employing any augmentation algorithms or complicated features-engineering tools.

Future work can consider further enhanced versions of the designed CNN algorithms that perform adequately with other highly imbalanced Android datasets. So, different Android datasets of new malware attack families can be examined and investigated. In addition, the authors intend to collect and build our Android dataset that composes ransomware attacks. This is to test further the classification efficiency of the developed CNN algorithms and their detection capabilities in identifying and recognizing different recent families of Android malware or ransomware attacks. Moreover, the authors intend to propose and develop an image-based real-time Android malware detection system in our future work. So, the authors have already started developing an Android mobile application and a cloud-based back-end web service that can detect malware APK files while downloading the APK files from the google store by first converting them into images.

ACKNOWLEDGMENT

The authors would like to thank the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication. Moreover, this research was done during the author Iman Almomani's sabbatical year 2021/2022 from The University of Jordan, Amman, Jordan.

REFERENCES

[1] *Statista Report Mobile Operating Systems' Market Share Worldwide From January 2012 to June 2021*. Accessed: Jul. 3, 2021. [Online]. Available: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

[2] S. Mahdavifar, A. F. Abdul Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "Dynamic Android malware category classification using semi-supervised deep learning," in *Proc. IEEE Int. Conf Dependable, Auton. Secure Comput., Int. Conf Pervasive Intell. Comput., Int. Conf Cloud Big Data Comput., Int. Conf Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Aug. 2020, pp. 515–522.

[3] S. Selvaganapathy, S. Sadasivam, and V. Ravi, "A review on Android malware: Attacks, countermeasures and challenges ahead," *J. Cyber Secur. Mobility*, vol. 10, pp. 177–230, Mar. 2021.

[4] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on autoencoders and API-images," *J. Parallel Distrib. Comput.*, vol. 137, pp. 26–33, Mar. 2020.

[5] I. Almomani, R. Qaddoura, M. Habib, S. Alsoghyer, A. A. Khayer, I. Aljarah, and H. Faris, "Android ransomware detection based on a hybrid evolutionary approach in the context of highly imbalanced data," *IEEE Access*, vol. 9, pp. 57674–57691, 2021.

[6] V. Kouliaridis and G. Kambourakis, "A comprehensive survey on machine learning techniques for Android malware detection," *Information*, vol. 12, no. 5, p. 185, Apr. 2021.

[7] I. Almomani, A. AlKhayer, and M. Ahmed, "An efficient machine learning-based approach for Android v.11 ransomware detection," in *Proc. 1st Int. Conf. Artif. Intell. Data Anal. (CAIDA)*, Apr. 2021, pp. 240–244.

[8] T. H.-D. Huang and H.-Y. Kao, "R2-D2: ColoR-inspired convolutional neural network (CNN)-based Android malware detections," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 2633–2642.

[9] H. Naeem, "Detection of malicious activities in Internet of Things environment based on binary visualization and machine intelligence," *Wireless Pers. Commun.*, vol. 108, no. 4, pp. 2609–2629, Oct. 2019.

[10] F. Ullah, H. Naeem, S. Jabbar, S. Khalid, M. A. Latif, F. Al-Turjman, and L. Mostarda, "Cyber security threats detection in Internet of Things using deep learning approach," *IEEE Access*, vol. 7, pp. 124379–124389, 2019.

[11] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Comput. Netw.*, vol. 171, Apr. 2020, Art. no. 107138.

[12] Z. Ren, H. Wu, Q. Ning, I. Hussain, and B. Chen, "End-to-end malware detection for Android IoT devices using deep learning," *Ad Hoc Netw.*, vol. 101, Apr. 2020, Art. no. 102098.

[13] W. Chao, L. Qun, W. XiaoHu, R. TianYu, D. JiaHan, G. GuangXin, and S. EnJie, "An Android application vulnerability mining method based on static and dynamic analysis," in *Proc. IEEE 5th Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, Jun. 2020, pp. 599–603.

[14] M. Ziadia, J. Fattahi, M. Mejri, and E. Pricop, "Smali+: An operational semantics for low-level code generated from reverse engineering Android applications," *Information*, vol. 11, no. 3, p. 130, Feb. 2020.

[15] M. Gonçalves and A. C. R. Paiva, "Reverse engineering of Android applications: REiMPAcT," in *Proc. Int. Conf. Quality Inf. Commun. Technol.* Faro, Portugal: Springer, 2020, pp. 369–382.

[16] M. A. Rahim Khan and M. K. Jain, "Protection Android app with multi-DEX and SO files from reverse engineering," *Mater. Today, Proc.*, pp. 1–9, Jan. 2021, doi: 10.1016/j.matpr.2020.12.190.

[17] B. A. Mantoo and S. S. Khurana, "Static, dynamic and intrinsic features based Android malware detection using machine learning," in *Proc. ICRIC*. Kashmir, India: Springer, 2020, pp. 31–45.

[18] P. Agrawal and B. Trivedi, "Machine learning classifiers for Android malware detection," in *Data Management, Analytics and Innovation*. Kuala Lumpur, Malaysia: Springer, 2021, pp. 311–322.

[19] I. Almomani and A. Khayer, "Android applications scanning: The guide," in *Proc. Int. Conf. Comput. Inf. Sci. (ICCIS)*, Apr. 2019, pp. 1–5.

[20] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient DenseNet-based deep learning model for malware detection," *Entropy*, vol. 23, no. 3, p. 344, Mar. 2021.

[21] H. Kim, K. Kim, J. Hong, J. Heo, and J. Kook, "EDAroid: An efficient dynamic analysis tool for Android applications," in *Proc. Int. Conf. Res. Adapt. Convergent Syst.*, Oct. 2020, pp. 261–266.

[22] I. Pustogarov, Q. Wu, and D. Lie, "Ex-vivo dynamic analysis framework for Android device drivers," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1088–1105.

[23] R. Thangavelooa, W. W. Jinga, C. K. Lenga, and J. Abdullaha, "DATDroid: Dynamic analysis technique in Android malware detection," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 10, no. 2, pp. 536–541, 2020.

[24] H. Hasan, B. T. Ladani, and B. Zamani, "MEGDroid: A model-driven event generation framework for dynamic Android malware analysis," *Inf. Softw. Technol.*, vol. 135, Jul. 2021, Art. no. 106569.

[25] A. Mohaisen, O. Alrawi, and M. Mohaisen, "AMAL: High-fidelity, behavior-based automated malware analysis and classification," *Comput. Secur.*, vol. 52, pp. 251–266, Jul. 2015.

[26] E. Amer and I. Zelinka, "A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence," *Comput. Secur.*, vol. 92, May 2020, Art. no. 101760.

[27] R. Sihwail, K. Omar, K. Z. Ariffin, and S. A. Afghani, "Malware detection approach based on artifacts in memory image and dynamic analysis," *Appl. Sci.*, vol. 9, no. 18, p. 3680, Sep. 2019.

[28] H. Naeem, F. Ullah, M. R. Naeem, S. Khalid, D. Vasan, S. Jabbar, and S. Saeed, "Malware detection in industrial Internet of Things based on hybrid image visualization and deep learning model," *Ad Hoc Netw.*, vol. 105, Aug. 2020, Art. no. 102154.

[29] L. N. Vu and S. Jung, "AdMat: A CNN-on-matrix approach to Android malware detection and classification," *IEEE Access*, vol. 9, pp. 39680–39694, 2021.

[30] Y. Ding, X. Zhang, J. Hu, and W. Xu, "Android malware detection method based on bytecode image," *J. Ambient Intell. Hum. Comput.*, vol. 11, pp. 1–10, Jun. 2020.

[31] A. Darwaish and F. Naït-Abdesselam, "RGB-based Android malware detection and classification using convolutional neural network," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–6.

[32] M. Kinkead, S. Millar, N. McLaughlin, and P. O'Kane, "Towards explainable CNNs for Android malware detection," *Proc. Comput. Sci.*, vol. 184, pp. 959–965, Jan. 2021.

[33] J. Singh, D. Thakur, F. Ali, T. Gera, and K. S. Kwak, "Deep feature extraction and classification of Android malware images," *Sensors*, vol. 20, no. 24, p. 7013, Dec. 2020.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.

[35] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[36] D. Theckedath and R. R. Sedamkar, "Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks," *Social Netw. Comput. Sci.*, vol. 1, no. 2, pp. 1–7, Mar. 2020.

[37] T. Carvalho, E. R. S. de Rezende, M. T. P. Alves, F. K. C. Balieiro, and R. B. Sovat, "Exposing computer generated images by eye's region classification via transfer learning of VGG19 CNN," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 866–870.

[38] F. Hong, C. Lu, W. Jiang, W. Ju, and T. Wang, "RDNet: Regression dense and attention for object detection in traffic symbols," *IEEE Sensors J.*, vol. 21, no. 22, pp. 25372–25378, Nov. 2021.

[39] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[40] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the ResNet model for visual recognition," *Pattern Recognit.*, vol. 90, pp. 119–133, Jun. 2019.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[42] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. de Geus, "Malicious software classification using transfer learning of ResNet-50 deep neural network," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 1011–1014.

[43] X. Ou, P. Yan, Y. Zhang, B. Tu, G. Zhang, J. Wu, and W. Li, "Moving object detection method via ResNet-18 with encoder–decoder structure in complex scenes," *IEEE Access*, vol. 7, pp. 108152–108160, 2019.
[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[45] Q. A. Al-Haija, M. Smadi, and O. M. Al-Bataineh, "Identifying phasic dopamine releases using DarkNet-19 convolutional neural network," in *Proc. IEEE Int. IoT, Electron. Mechatronics Conf. (IEMTRONICS)*, Apr. 2021, pp. 1–5.
[46] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.
[47] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1452–1464, Jun. 2018.
[48] F. Saxen, P. Werner, S. Handrich, E. Othman, L. Dinges, and A. Al-Hamadi, "Face attribute detection with MobileNetV2 and NasNet-mobile," in *Proc. 11th Int. Symp. Image Signal Process. Anal. (ISPA)*, Sep. 2019, pp. 176–180.
[49] R. U. Khan, X. Zhang, and R. Kumar, "Analysis of ResNet and GoogleNet models for malware detection," *J. Comput. Virol. Hacking Techn.*, vol. 15, no. 1, pp. 29–37, 2019.
[50] H. Lee, I. Ullah, W. Wan, Y. Gao, and Z. Fang, "Real-time vehicle make and model recognition with the residual SqueezeNet architecture," *Sensors*, vol. 19, no. 5, p. 982, Feb. 2019.
[51] N. A. El-Hag, A. Sedik, W. El-Shafai, H. M. El-Hoseny, A. A. Khalaf, A. S. El-Fishawy, W. Al-Nuaimy, F. E. A. El-Samie, and G. M. El-Banby, "Classification of retinal images based on convolutional neural network," *Microsc. Res. Tech.*, vol. 84, no. 3, pp. 394–414, 2021.
[52] I. K. M. Jais, A. R. Ismail, and S. Q. Nisa, "Adam optimization algorithm for wide and deep neural network," *Knowl. Eng. Data Sci.*, vol. 2, no. 1, pp. 41–46, 2019.
[53] H. Gao, Y. Yang, S. Lei, C. Li, H. Zhou, and X. Qu, "Multi-branch fusion network for hyperspectral image classification," *Knowl.-Based Syst.*, vol. 167, pp. 11–25, Mar. 2019.
[54] T. Hegazy, P. Fazio, and O. Moselhi, "Developing practical neural network applications using back-propagation," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 9, no. 2, pp. 145–159, Mar. 1994.
[55] M. Stamp, M. Alazab, and A. Shalaginov, *Malware Analysis Using Artificial Intelligence and Deep Learning*. Switzerland: Springer, 2021.
[56] A. P. Namanya, I. U. Awan, J. P. Disso, and M. Younas, "Similarity hash based scoring of portable executable files for efficient malware detection in IoT," *Future Gener. Comput. Syst.*, vol. 110, pp. 824–832, Sep. 2020.

IMAN ALMOMANI (Senior Member, IEEE) received the bachelor's degree from United Arab Emirates, in 2000, the master's degree in computer science from Jordan, in 2002, and the Ph.D. degree in wireless network security from De Montfort University, U.K., in 2007. She is currently an Associate Professor in cybersecurity. She is the Associate Director with the Research and Initiatives Centre (RIC) and also the Leader with the Security Engineering Laboratory (SEL), Prince Sultan University (PSU), Riyadh, Saudi Arabia. Before Joining PSU, she worked as an Associate Professor and the Head of the Computer Science Department, The University of Jordan, Jordan. Her research interests include wireless networks and security, mainly wireless mobile *ad-hoc* networks (WMANETs), wireless sensor networks (WSNs), multimedia networking (VoIP), and security issues in wireless networks. She is also interested in the area of electronic learning (e-learning) and mobile learning (m-learning). She has several publications in the above areas in a number of reputable international and local journals and conferences. She is in the organizing and technical committees for a number of local and international conferences. She is also a Senior Member of IEEE WIE. Also, she serves as a reviewer and a member for the editorial board in a number of international journals.

AALA ALKHAYER received the Bachelor of Engineering degree in information technology engineering from SVU University, Damascus, in 2017, and the bachelor's degree in software engineering from Prince Sultan University (PSU), Riyadh, Saudi Arabia, in 2018. She is currently a Research Engineer at the Security Engineering Laboratory (SEL), PSU. Her research interests include software engineering, networks security, malware analysis, multimedia networking, and computer vision.

WALID EL-SHAFAI was born in Alexandria, Egypt. He received the B.Sc. degree (Hons.) in electronics and electrical communication engineering from the Faculty of Electronic Engineering (FEE), Menoufia University, Menouf, Egypt, in 2008, the M.Sc. degree from the Egypt-Japan University of Science and Technology (E-JUST), in 2012, and the Ph.D. degree from the Faculty of Electronic Engineering, Menoufia University, in 2019. Since January 2021, he has been joined as a Postdoctoral Research Fellow at the Security Engineering Laboratory (SEL), Prince Sultan University (PSU), Riyadh, Saudi Arabia. He is currently working as a Lecturer and an Assistant Professor with the Electronics and Communication Engineering (ECE) Department, FEE, Menoufia University. His research interests include wireless mobile and multimedia communications systems, image and video signal processing, efficient 2D video/3D multi-view video coding, multi-view video plus depth coding, 3D multi-view video coding and transmission, quality of service and experience, digital communication techniques, cognitive radio networks, adaptive filters design, 3D video watermarking, steganography, and encryption, error resilience and concealment algorithms for H.264/AVC, H.264/MVC, and H.265/HEVC video codecs standards, cognitive cryptography, medical image processing, speech processing, security algorithms, software defined networks, the Internet of Things, medical diagnoses applications, FPGA implementations for signal processing algorithms and communication systems, cancellable biometrics and pattern recognition, image and video magnification, artificial intelligence for signal processing algorithms and communication systems, modulation identification and classification, image and video super-resolution and denoising, cybersecurity applications, malware and ransomware detection and analysis, deep learning in signal processing, and communication systems applications. He has several publications in the above research areas in several reputable international and local journals and conferences. Also, he serves as a reviewer for several international journals.