Hindawi
Security and Communication Networks
Volume 2022, Article ID 8893764, 12 pages
https://doi.org/10.1155/2022/8893764

Research Article

Android Malware Detection Technology Based on Lightweight Convolutional Neural Networks

Genchao Ye,[1,2] **Jian Zhang**,[1,2] **Huanzhou Li**,[1,2] **Zhangguo Tang**,[1,2] **and Tianzi Lv**[1,2]

[1]*School of Physics and Electronic Engineering, Sichuan Normal University, Chengdu 610101, China*
[2]*Institute of Network and Communication Technology, Sichuan Normal University, Chengdu 610101, China*

Correspondence should be addressed to Jian Zhang; zhangjian@sicnu.edu.cn

Received 22 May 2021; Revised 9 July 2021; Accepted 23 February 2022; Published 16 March 2022

Academic Editor: Usman Habib

With the rapid development of Android, a major mobile Internet platform, Android malware attacks have become the number one threat to mobile Internet security. Traditional malware detection methods have low precision and greater time complexity. At present, image detection methods based on deep learning are used in malware detection. However, most of these methods are based on the largescale convolutional neural network model (such as VGG16). The computation and weight files of these models are very large, so they are not suitable for mobile Internet platforms with limited computation. A novel detection method based on a lightweight convolutional neural network is presented in this study. It transforms Android malware classes.dex, Android-manifest.xml, and resource.arsc into RGB images and uses the lightweight convolutional neural network to extract the features of RGB images automatically. The experimental results of this study indicate that the method performs well in terms of precision and speed of detection.

1. Introduction

Android is one of the most common smartphone operating systems. According to the statistics of the global smartphone market system occupancy in the second quarter of 2018 by Kantar, an international data research organization, 82% of the market share was occupied by Android, and the share is still increasing. The rapid popularization of Android has not only brought great convenience to users but also attracted the attention of network hackers. Due to the openness of Android, Android devices are susceptible to malware infection. AV-TEST Security Report 2019/2020 [1] shows that 6,201,358 new Android malware samples were added in 2017, and the number of new malware samples has increased from 2019. In addition, Android won the first place in 2019 with 417 known security vulnerabilities in the ranking of operating systems and programs with the number of security vulnerabilities and was listed in the common vulnerabilities and exposure (CVE) database. Among all Android malware in 2019, Trojan accounted for more than 90%, followed by malicious adware and ransomware, which are also the main ways for criminals to illegally obtain benefits.

Obviously, it is of extreme urgency to detect and prevent malware in order to resolve serious security problems caused by Android malware. In the past, static analysis technology and dynamic analysis technology [2] were the main methods of Android malware detection. Static analysis [3] refers to the extraction and analysis of the features of the software without running. Lei [4] extracted the API call sequence of the Android software through reverse engineering and normalization and then detected it through a probability discriminant learning model based on regular logistic regression. MaMaDroid [5] constructed the behavior model in the form of Markov chain for the API call sequence and classified malware. Tian [6] solved the problem of repackaged malware detection by analyzing the dependence between classes and classes, methods, and methods according to the heterogeneity of Android malware code. Kong [7] extracted structural information on the malware program as an attribute function call graph and then used an integrated classifier for automatic malware classification. However, it is

very difficult to the reverse engineering of the Android software for static analysis, and it is easy to be interfered by code obfuscation technology or encryption technology, resulting in inaccurate detection results. Dynamic analysis technology [8] refers to executing malware in a secure virtual environment and monitoring and extracting malware behavior information dynamically. The MADAM system [9] could judge whether it is malicious according to the seven abnormal behaviors of the software when it is running. Yuan [10] performed a dynamic taint analysis through system hooks and monitored various application operations. Each behavior feature was one-hot encoded and transmitted to the Deep Belief Networks (DBN) as trained data. DroidTrace [11] was a dynamic analysis system with forward execution capability based on the Ptrace which is a dynamic analysis tool. It could judge whether it is malware based on the system dynamic load when the software is running. Somarriba [12] provided a framework of monitoring and visualized abnormal function calls of Android applications. Dynamic analysis technology has a high accuracy rate, but repeated code execution is time-consuming and resource intensive.

In recent years, visualization technology has been introduced into malware detection. Malware visualization technology refers to converting some elements of malware into images' form for detection. In 2011, Nataraj et al. [13] first proposed to visualize binary files of computer malware as grayscale images and then to classify the malware by using the image features. Mclaughlin et al. [14] extracted the opcode sequence of the code instructions in the Android software and mapped it in the vector space and then used CNN to achieve detection. Kumar et al. [15] transformed malware into grayscale scale images and then detected and classified the malware by Random Forest algorithm (RF). On this basis, Darus et al. [16] compared the accuracy of the K-Nearest Neighbor algorithm (KNN), RF, and Decision Tree algorithm (DT) applied to malware grayscale images' classification and detection. And they found out the one with the best performance among the three algorithms is RF. Poonguzhali et al. [17] proposed to combine convolutional neural network and intelligent optimization algorithm for malware image detection and achieved high detection accuracy. Many research based on visualization technology has been implemented in computer malware detection, but few on Android malware detection. The detailed analysis of the above references can be found in Table 1.

There are a lot of types and variations of Android malware; their variations and code obfuscation, shell and encryption, make detection more difficult. Traditional convolutional neural network for Android has many parameters and a large amount of calculation, which leads to a long training time and is not suitable for the mobile Internet platform. For the reasons mentioned above, this study proposes to convert Android malware to RGB images and to train and detect it by depthwise separable convolutional neural networks, which improves the training speed and reduces the number of parameters while having fine accuracy.

The rest of this paper is organized as follows. Section 2 describes the basic principle of convolution neural network and depthwise separable convolution. In Section 3, it introduces the method of this study which includes a new malware visualization method and MobileNet V2 network model. Experimental results and performance analysis are discussed in Section 4. And the conclusion of this study is presented in Section 5.

2. Depthwise Separable Convolutional Neural Networks

2.1. Deep Learning and Convolutional Neural Networks. Deep learning [18] is a collection of algorithms that use various machine learning algorithms to solve various problems such as images and texts on multilayer neural networks. It can provide feature learning and obtain higher-level abstract features that human may not recognize. These abstract features can help to capture relevant characteristics. Among deep learning models, convolutional neural networks [19] have outstanding performance in the field of image recognition. Compared with traditional machine learning and other models, CNN has a strong ability to obtain higher-level features of the images. A CNN includes the input layers, the convolution layers, the ReLU excitation layers, the pooling layers, and the fully connected layers. The current common CNN models are VGGNet, ResNet, GoogleNet, and MobileNet.

2.2. Standard Convolution. Standard convolution uses N $D_k \times D_k$ convolution kernels to perform convolution calculations on three-channel RGB images and finally output N Feature Maps. The process is shown in Figure 1.

The number of parameters and calculation cost of standard convolution are described as follows:

$$\begin{aligned} P_1 &= D_k \times D_k \times M \times N, \\ C_1 &= D_k \times D_k \times M \times N \times D_F \times D_F, \end{aligned} \tag{1}$$

where $D_k \times D_k$ represents the size of the convolution kernel, $D_k \times D_k$ *is* the size of the feature maps, and M and N denote the number of input channels and output channels, respectively.

2.3. Depthwise Separable Convolution. Depthwise separable convolution [20]-based MobileNet v2 model is used in this study. Compared with standard convolution, this model has faster calculation speed and lower computational cost. Depthwise separable convolution is a combination of depthwise convolution (DW) and pointwise convolution (PW), in which the number of parameters and calculation cost of extracting feature are lower than standard convolution.

(1) Depthwise convolution: after inputting a three-channel RGB image, DW will perform the first convolution operation, which is completely performed in a two-dimensional plane. The number of convolution kernels is the same as the number of channels of the input image. Therefore, a three-

Table 1: Analysis of related works.

Types	References	Target features	Methods	Analysis
Static detection	[4]	API call sequence	Decompiling the source code to extract features and detecting them by the probability discriminant model	The pretreatment process is complex
	[6]	Classes and class dependencies	Analyzing the heterogeneity of android malware code by machine learning algorithm	Model training takes a long time
	[7]	Functions call sequence	Computing malware similarity through machine learning algorithm	The accuracy is not high
Dynamic detection	[9]	Malicious behavior	It can be judged by the abnormal behavior of software running	Serious memory consumption
	[10]	Malicious behavior	The behavior features are transformed into hot coding and transmitted to the DBN network for training	The preprocessing is complex and the trigger mechanism needs to run the system all the time
	[11]	System call behavior	Monitor and analyze system call behavior from time to time	The trigger mechanism needs to run the system all the time
	[12]	API functions	Introducing hooks in order to trace restricted API functions used at runtime of the application	The system starts slowly and occupies a large proportion of memory
Visualization detection	[13]	Binary files	Malware binaries are visualized as grayscale images and classified by analyzing the image feature descriptor	The feature of the grayscale image is single and model training takes a long time
	[15]	.Dex files	The .dex files are transformed into RGB images and classified by random forest algorithm	The accuracy is not high
	[17]	Binary files	Converting the malicious code into the grayscale images, these images were identified and classified by the convolutional neural networks; then, using the bat algorithm to eliminate over fitting of the model over the fitting and the under-fitting	The feature of the grayscale image is single and model training takes a long time

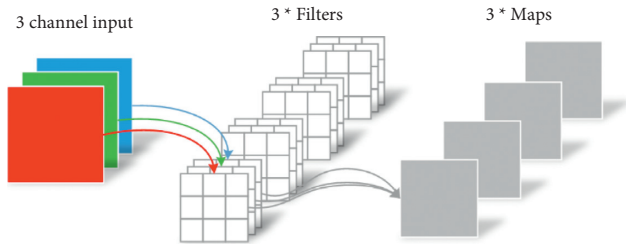


Figure 1: Standard convolution process.

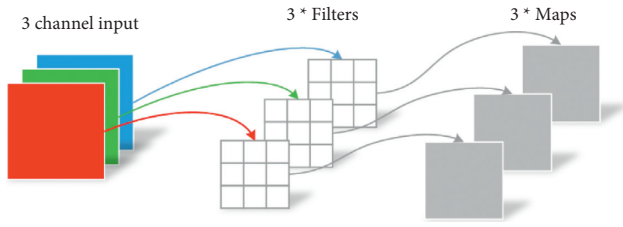


Figure 2: Depthwise convolution process.

channel image generates three feature maps after calculation, as shown in Figure 2.

(2) Pointwise convolution: the operation of the PW is similar to the standard convolution operation, as shown in Figure 3. The size of its convolution kernel is $1 \times 1 \times M$, and M is the number of channels in the previous layer. This convolution will weight and combine the previous maps in the depth direction to generate new feature maps.

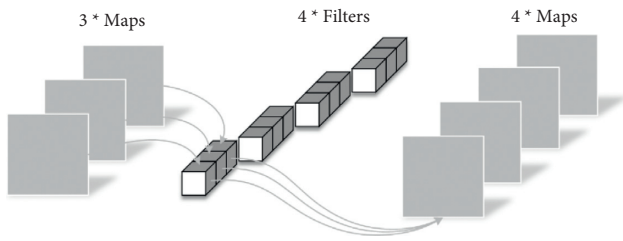


Figure 3: Pointwise convolution process.

The number of parameters and calculation cost of the depth separable convolution are represented as follows:

$$C_2 = D_k \times D_k \times M \times D_F \times D_F + M \times N \times D_F \times D_F,$$
$$P_2 = M \times D_k \times D_k + M \times N. \quad (2)$$

The results of the depthwise convolution and pointwise convolution mentioned above are the same as the result of the standard convolution. Equations (3) and (4) are the ratios of the number of parameters and the calculation cost of the depthwise separable convolution and the standard convolution under the same effect. It can be seen that the number of parameters and the calculation cost of the depthwise separable convolution are greatly reduced. The size of the convolution kernel is set as 3×3 usually. In this case, the number of

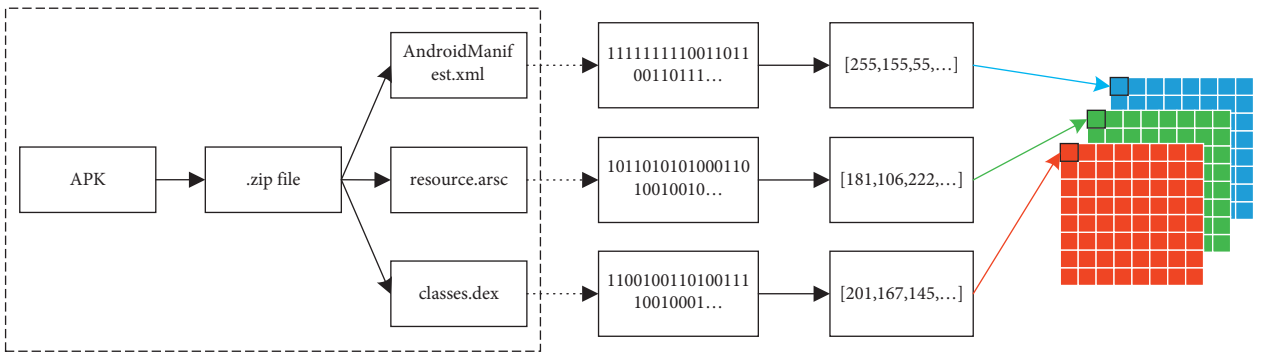


Figure 4: Visualization process of Android software.

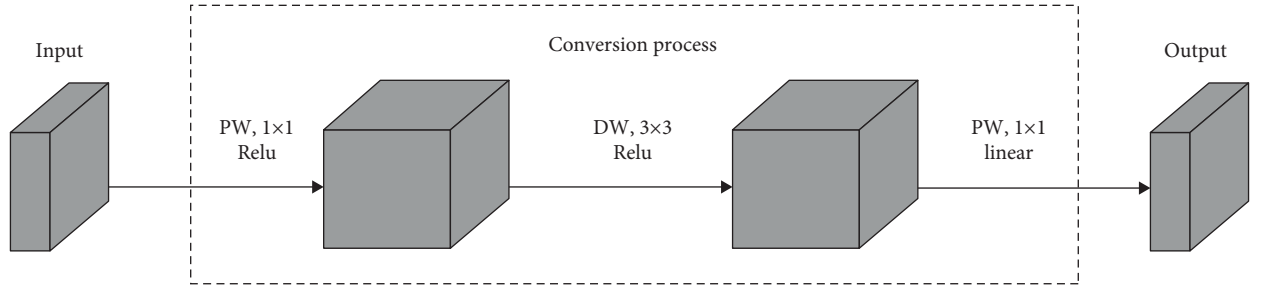


Figure 5: Convolution process of inverse residual depthwise separable convolution.

Table 2: Comparison of CNN model parameters.

Model name	Number of parameters
VGG16	138342976
Inception v3	24734048
ResNet	23518273
MobileNet v1	4209088
MobileNet v2	2958440

Table 3: Classification of dataset (a) and (b).

Dataset	Category	Quantity
(a)	Benign	13500
	Malware	13787
(b)	Adware	1515
	Banking	2506
	Benign	4942
	Riskware	4362
	SMS malware	4822

parameters and calculation cost of the depthwise separable convolution are about 1/9 of those of the standard convolution:

$$\frac{P_2}{P_1} = \frac{D_k \times D_k \times M + M \times N}{D_k \times D_k \times M \times N} = \frac{1}{N} + \frac{1}{D_k^2}, \tag{3}$$

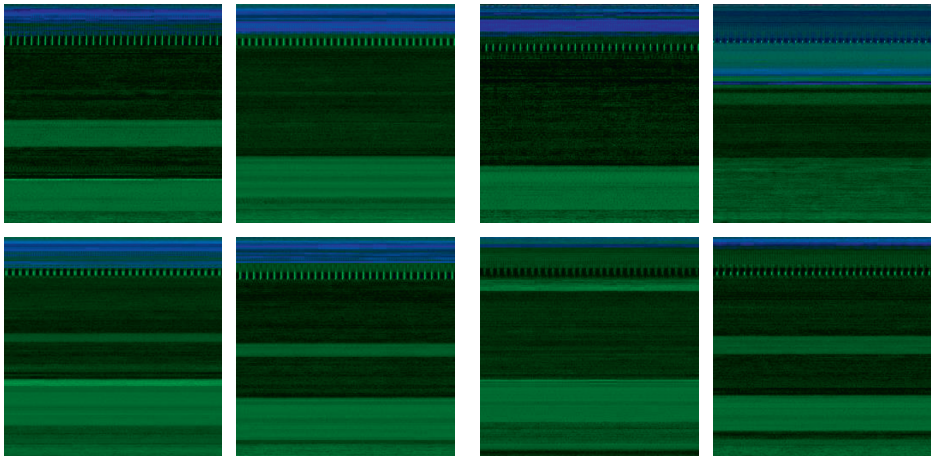
$$\frac{C_2}{C_1} = \frac{D_k \times D_k \times M \times D_F \times D_F + M \times N \times D_F \times D_F}{D_k \times D_k \times M \times N \times D_F \times D_F} = \frac{1}{N} + \frac{1}{D_k^2}. \tag{4}$$

3. Methods

In this section, this study primarily proposes to visualize Android software as RGB images and then detects it through the lightweight convolution neural network.

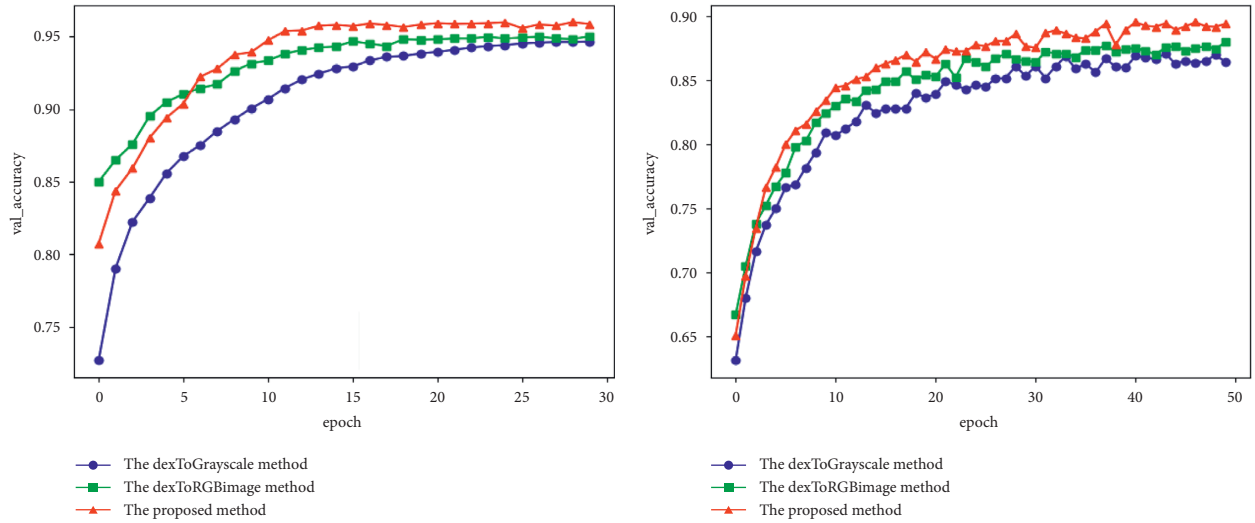
3.1. Visualization Process. Android application package (APK) [21] is a file format used by the Android system to install applications (APP), and the suffix is .apk. In fact, the APK can be decompressed by changing the suffix name to .zip, which contains META-INF folder, res folder, resources.arsc, AndroidManifest.xml, and classes.dex. Among them, classes.dex is an 8 bit binary file, which is the logical part of the Android software to realize functions. Resources.arsc is the compiled binary resource file. AndroidManifest.xml contains the configuration information of the APP.

This study proposes to convert Android software to RGB images for detection. RGB [21] images are composed of a three-dimensional array of the $M \times N \times 3$ format, which can also be understood as being constructed by three $M \times N$ grayscale images. The three images represent the three components of R, G, and B. And the pixel point of each component ranges from [0, 255]. From this, the value expression of RGB image pixel points is (R: [0, 255], G: [0, 255], B: [0, 255]). The visualization process is shown in Figure 4. The suffix name of the APK is changed to .zip file. Then, the classes.dex, resources.arsc, and AndroidManifest.xml are extracted by the zip file library of the python. The decimal range of each byte is in [0, 255],



(a) (b)

Figure 6: Some RGB images of (a) benign software and (b) malware.



(a) (b)

Figure 7: Training process of visualization methods.

which is exactly the same as the value range of each component pixel of the RGB images. So, these three files are filled with channels of *R*, G, and B to generate RGB images by the PIL library.

3.2. Model Summary. The model named MobileNet v2 [22] implemented in this study is a lightweight CNN model proposed by Google in 2018, which uses depthwise separable convolution. MobileNet v2 is an improvement in MobileNet v1. MobileNet v1 is only a stack of depthwise separable convolutions; however, MobileNet v2 introduces a residual structure to improve the performance of the network. In addition, in order to preserve the diversity of features, the nonlinear activation function of ReLU is not adopted at the end of each depthwise separable convolution. The structure of MobileNet includes DW and PW. Before DW, MobileNet v2 adds a PW for performing dimension upgrade. This method makes DW work in higher dimensions, which can get a better effect. The convolution structure of MobileNet v2 is shown in Figure 5.

Table 2 shows the number of parameters of MobileNet v2 and other models. It can be seen that MobileNet v2 has fewer parameters and faster training speed than MobileNet v1, so this study uses MobileNet v2 for Android malware detection.

3.3. Functions of Training Process. Android malware detection is a two-classification problem. Sigmoid which is a good threshold function is used as the output of the classifier. It can map the output variable to (0, 1). If the output variable is greater than 0.5, it is considered to belong to one category; otherwise, it belongs to another category. The formula for the sigmoid function is as follows:

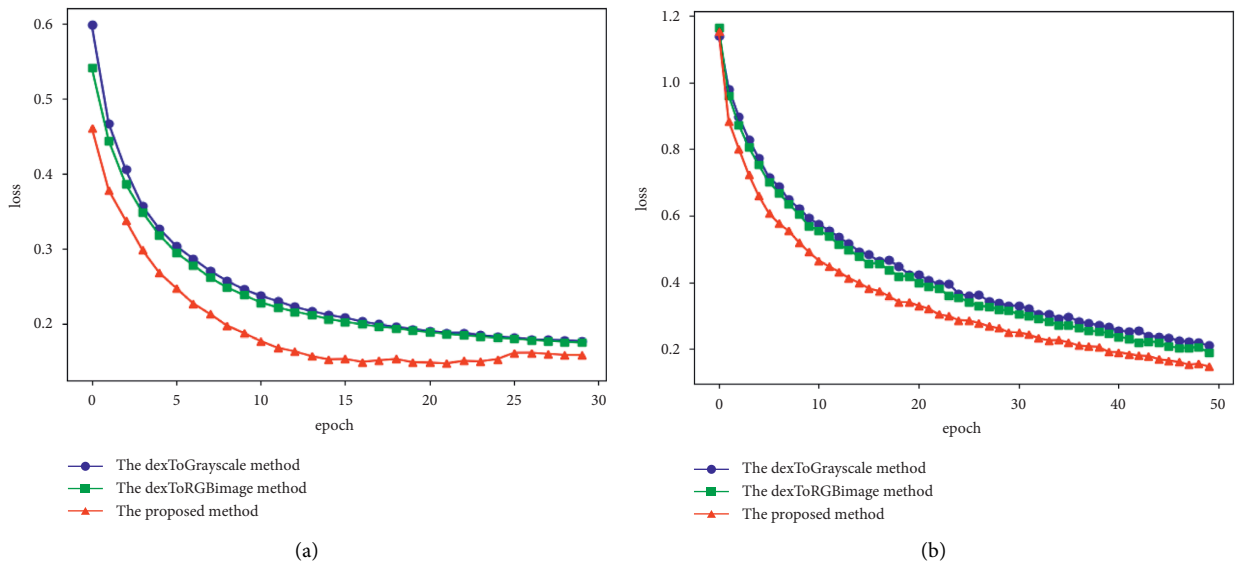


Figure 8: Loss-epoch curves of visualization methods.

Table 4: Comparison of accuracy and training time of visualization methods.

Dataset	Neural networks' models	Visualization methods	Val_accuracy (%)	Training time (1 epochs)
((s)a)	MobileNet v2	dexToGrayscale	94.44	42.02
		dexToRGBimage	95.01	44.41
		The proposed method	**95.98**	46.67
(b)	MobileNet v2	dexToGrayscale	87.09	39.99
		dexToRGBimage	87.97	40.18
		The proposed method	**89.54**	41.44

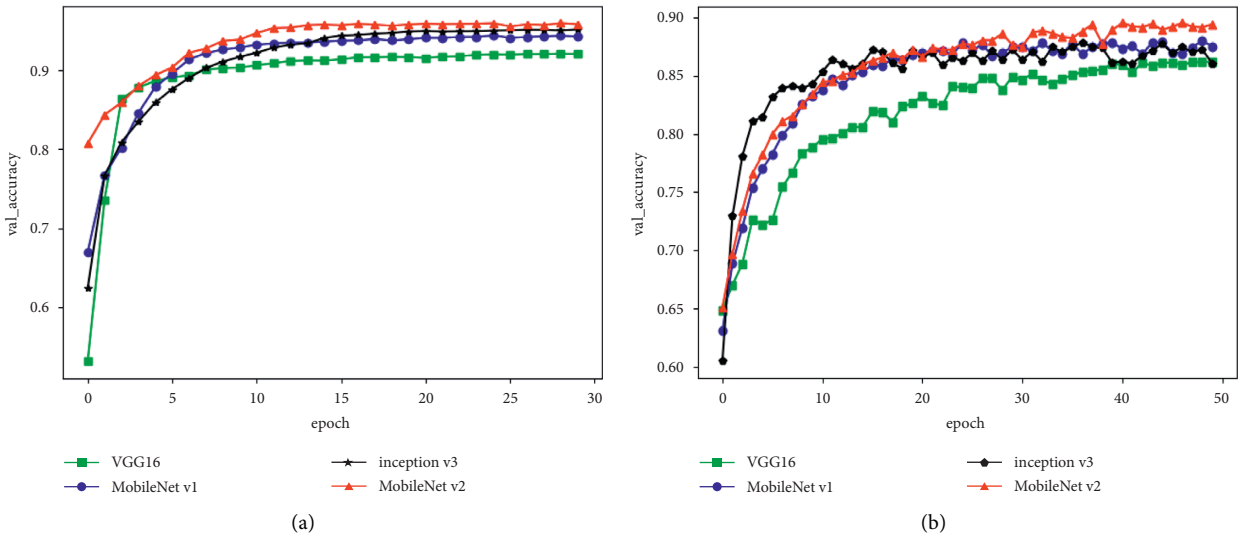


Figure 9: Training process of CNN models.

$$sigmoid(x) = \frac{1}{1+e^{-x}}, \tag{5}$$

where x represents the score of the sample of this category in the neural network.

Binary_cross entropy is used as the loss function in the training process of CNN, which is described as follows:

$$loss = -\sum_{i=1}^{n} \hat{y}_i \log\ y_i + (1-\hat{y}_i)\log(1-y_i), \tag{6}$$

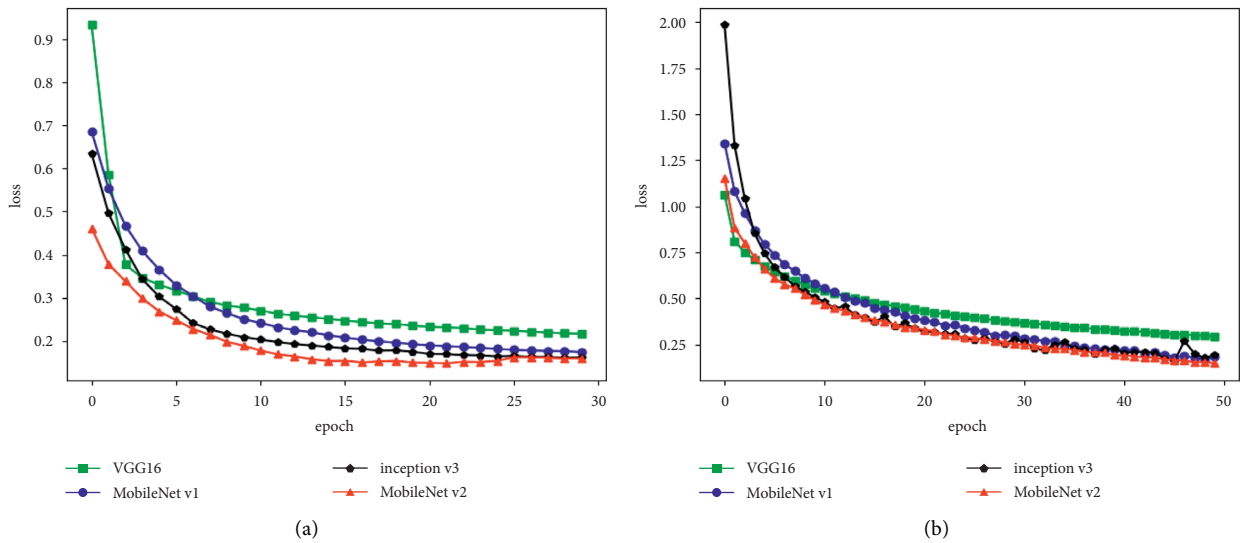


Figure 10: Loss-epoch curves of visualization methods.

Table 5: Comparison of accuracy and training time of CNN models.

Dataset	Visualization methods	Neural network models	Val_accuracy (%)	Training time (1 epochs)
((s)a)	**The proposed method**	Model 1: VGG16 [17]	92.12	74.11
		Model 2: MobileNet v1	94.39	46.03
		Model 3: Inception v3 [23]	95.19	62.55
		The proposed model: MobileNet v2	**95.98**	46.67
(b)	**The proposed method**	Model 1: VGG16 [17]	86.22	71.01
		Model 2: MobileNet v1	88.01	40.87
		Model 3: Inception v3 [23]	87.78	58.14
		The proposed model: MobileNet v2	**89.54**	41.44

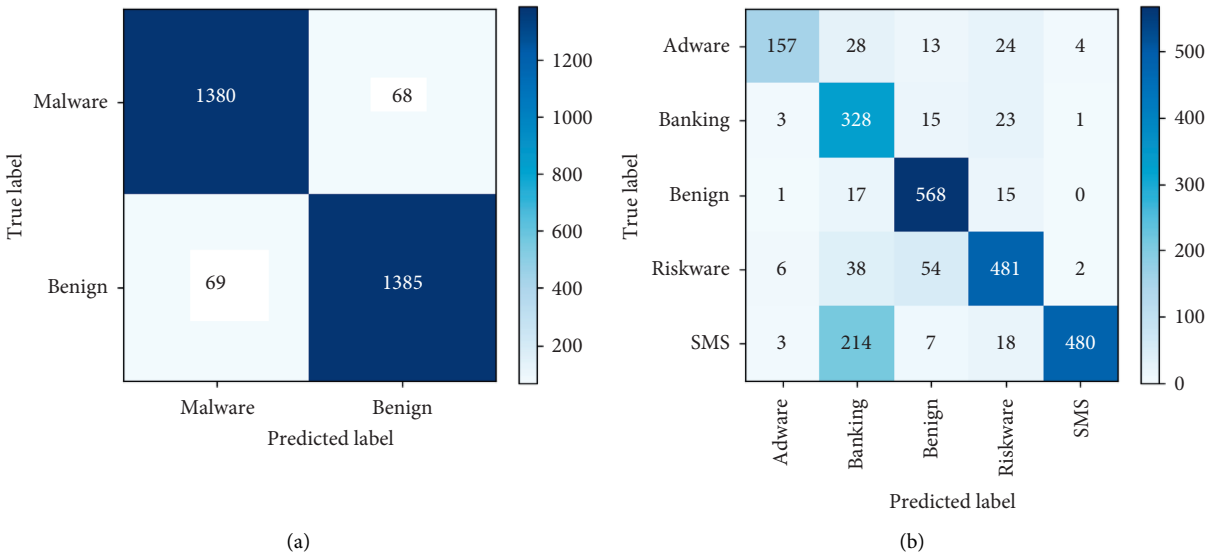


Figure 11: Confusion matrix for dataset (a) and (b).

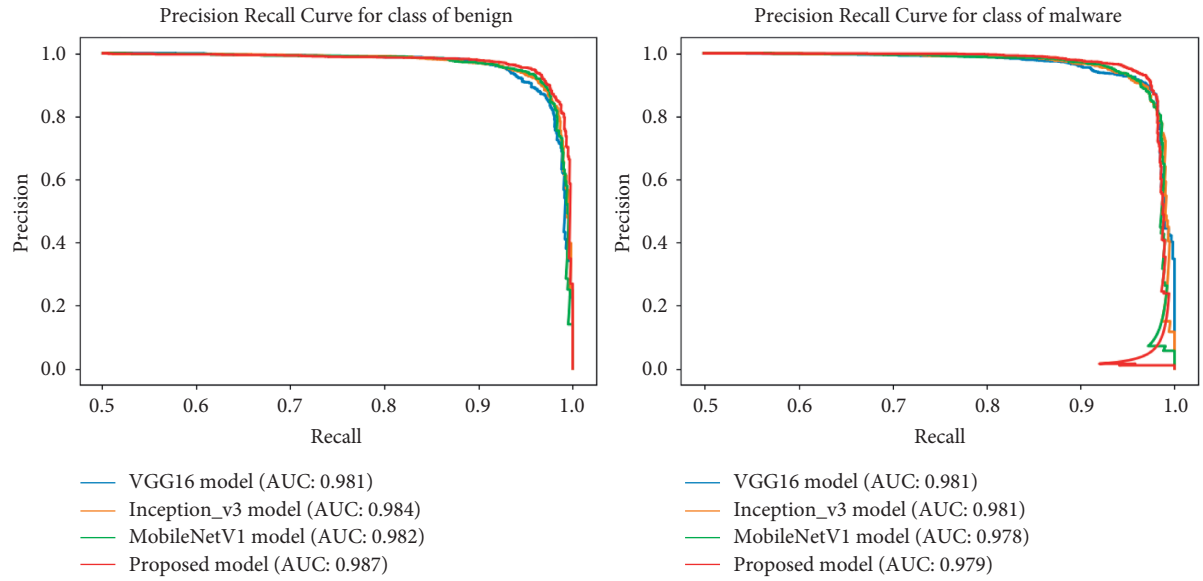


Figure 12: The P-R curves of dataset (a).

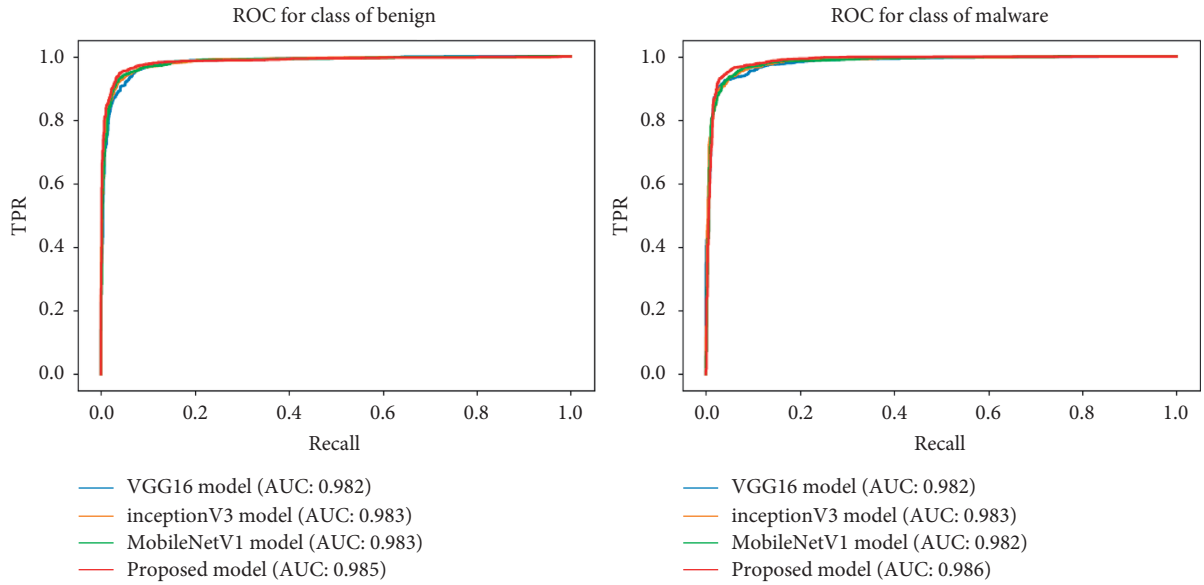


Figure 13: The ROC curves of dataset (a).

where $\widehat{y}_i$ is the true label value (0 or 1) and y_i is the predicted value given by the neural network model. At the end of each single training session, the neural networks will calculate the loss and then update the model parameters according to the loss.

4. Experiments and Discussion

This experiment was carried out under Ubuntu 18.04 version, the CPU is 8-core E5-2687W, the memory is 16 GB, the GPU is NVIDIA RTX 2080 TI, and the memory is 11 GB. This study use python and Tensorflow2.0 framework to design CNN models.

4.1. Dataset. The following two datasets were used in this experiment, and the details are shown in Table 3.

(1) Dataset (a) contains CIC-AndMal-2017, CIC-AndMal-2020, and 3000 benign Android software collected by our group. This dataset is divided into benign software and malware.

(2) Dataset (b) is CIC-AndMal-2020, which contains 5 types of Android software.

The dataset of this experiment has not been found used in other documents before. And this dataset is adopted in this study to compare the performance of the methods based on different visualization methods and different CNN models.

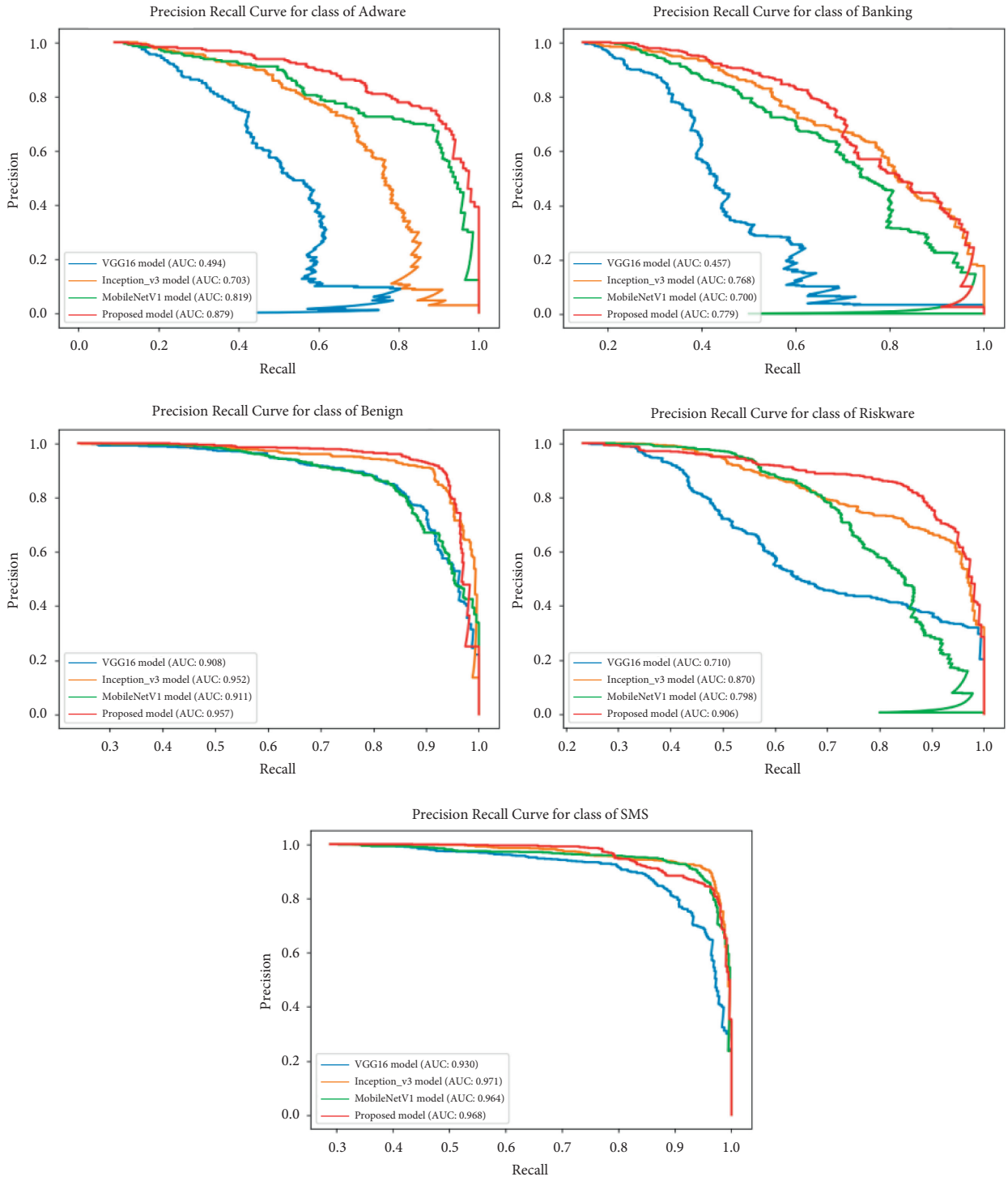


Figure 14: The P-R curves of dataset (b).

4.2. Comparison of Visualization Methods. In this study, dataset (a) and dataset (b) are separately divided into a training set and a test set of eight to two, and all samples are visualized to RGB images by the method mentioned in Section 3.1. Then, all images are scaled to the same size. Figure 6 shows some 256×256 RGB images generated by randomly selected Android benign software and malware. The difference in image texture between benign software and malware is very minor. However, the depthwise separable convolutional neural networks have a strong ability to image recognition and can judge whether it is malware through the subtle difference in the images.

The five-fold cross-validation method which can improve the accuracy and stability of the model is used in this experiment. The network parameters are initialized by MobileNet v2. Except for the last 10 layers of training, all the

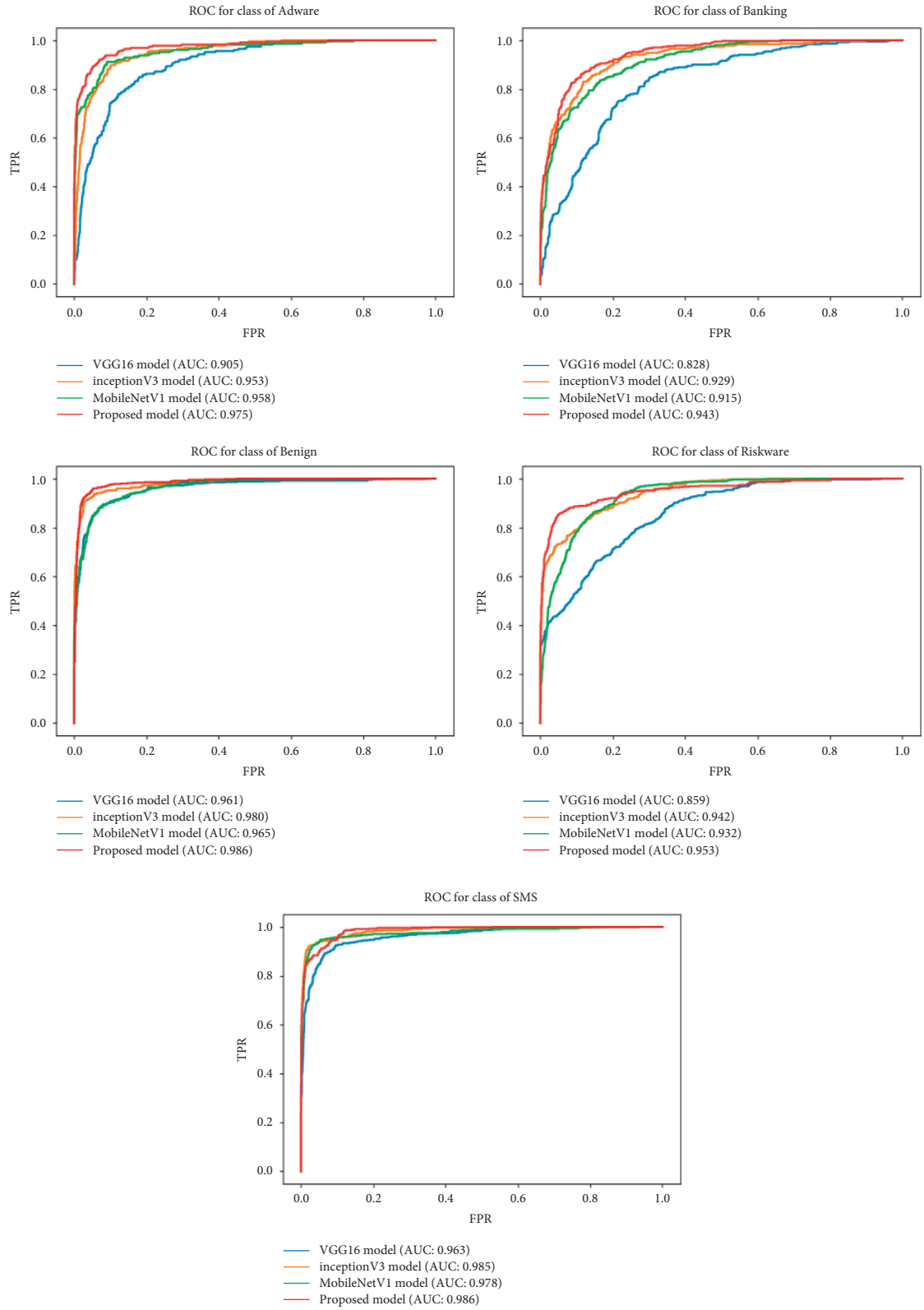


Figure 15: The ROC curves of dataset (b).

front layers are frozen. In the training process, the initial learning rate is set to 0.00001 and BATCH_SIZE is set to 32. The epoch is set to 30 in the experiment of dataset (a) and 50 for dataset (b).

Traditional visualization methods of Android software include dexToGrayscale and dexToRGBimage. They refer to the conversion of classes. dex files in APK into grayscale images or RGB images. The RGB images which are generated

by the classes.dex, resources.arsc, and AndroidManifest.xml can be easily distinguished in convolutional neural networks. The proposed visualization method is compared with the traditional visualization methods in the MobileNet v2 network model. The experiment is shown in Figures 7 and 8 and Table 4.

According to the results, the dexToGrayscale method needs more epochs to steady state and has the lowest accuracy. The effect of the dexToRGBimage method is better than the dexToGrayscale method. And the proposed method has the highest accuracy in the similar training time.

4.3. Comparison of Models. In this study, the CNN models in the previous document are compared with the MobileNet v2 used in this project, such as VGG16, Inception v3, and MobileNet v1.

The training process for dataset (a) and (b) is shown in Figures 9 and 10. The accuracy and the training time of the four models under the same conditions can be seen from Table 5.

In the training of dataset (a), the VGG16 model gets the lowest accuracy of 92.12% and the slowest training speed rate of 74.11 seconds at 1 epoch. The MobileNet v1 has fast training spend rate and high accuracy. And the Inception v3 is the opposite of it. The proposed method with MobileNet v2 has the accuracy of 95.98% and the training speed rate of 41.44 seconds at 1 epoch, which gets the highest accuracy in a relatively short time and has a certain improvement compared with the methods in historical document. Training result of dataset (b) is like that of dataset (a), MobileNet v2 takes the accuracy rate of 89.54%, and the training speed rate is 41.44 seconds at 1 epoch.

4.4. Model Analysis. Figure 11 shows the confusion matrix for dataset (a) and (b). In dataset (a), the true positive rate and false negative rate reached 99.3% and 95.2%, respectively. Among them, the recall rate of malicious software is 95.1%. In dataset (b), because of the small number of samples, many kinds and data imbalance, detection results of adware and SMS are not very good, but the true positive rate of banking malware, benign software, and riskware achieved 88.6%, 94.5%, and 82.8%, respectively.

Figures 12 and 13 show the P-R curves of dataset (a) and (b). The P-R curve is drawn under certain some thresholds based on precision rate and recall rate, which is an evaluation indicator that shows the quality of the model. In dataset (a), the AUC of malware and benign is 0.978 and 0.982, respectively. In another dataset, AUC of the five categories is 0.839, 0.779, 0.957, 0.906, and 0.968.

Figures 14 and 15 show the ROC curves of dataset (a) and (b). The abscissa of the ROC curve is the false positive rate and the ordinate is true positive rate. And ROC curve is a standard to measure the quality of a classification model. As can be seen from the figures, the model performs better in the margin class and SMS class because the number of other three kinds of data is small. Overall, the MobileNet V2 model used in this study has good performance and stability.

4.5. Discussion. Through the above experiments, it can be seen that the malware visualization method introduced in this study is better than the methods in [17, 23], etc. And its generalization ability is good. The performance and efficiency of MobileNet V2 in this experiment are also superior to those of the conventional convolutional neural network model. However, the model memory consumption of MobileNet V2 is still large. For the forthcoming period, we will consider compressing the network structure by referring to other lightweight network models, such as ShuffleNet [24], or optimizing the feature quantity by combining three feature files to generate a Markov graph [25] to further improve the performance of the detection method.

5. Conclusions

Android malware is continuously causing safety hazards to people's life, so the detection of Android malware is very necessary. This study proposed an Android malware detection method based on depthwise separable convolutional neural networks. The proposed algorithm converts classes.dex, resources.arsc, and AndroidManifest.xml of the Android malware samples to the RGB image. Compared with the grayscale image, RGB image possesses better color and texture characteristics. The quality and accuracy of MobileNet v2 model used in this experiment are higher than other models. Moreover, the number of parameters and calculation cost of MobileNet v2 are far lower than other models. If the model is deployed on Android phones, it can achieve a relatively outstanding malware detection function of the case of occupying a small amount of memory. So, this model is more suitable for Android malware detection. The significant improvement of the proposed method has been demonstrated by comparing it with the methods of some historical research.

Data Availability

Restrictions are applied to the availability of these data. Data were obtained from Canadian Institute for Cybersecurity and are available at https://www.unb.ca/cic/datasets/andmal2020.html with the permission of Canadian Institute for Cybersecurity.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] The AV-Test, "Facts & Analyses on the Threat Scenario: The AV-TEST Security Report 2019/2020," pp. 4–10, 2020, https://www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2019-2020.pdf.

[2] Q. Han, V. S. Subrahmanian, and Y. Xiong, "Android malware detection via (somewhat) robust irreversible feature transformations," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3511–3525, 2020.

[3] Y. Feng, S. Anand, I. Dillig, and A. Aiken, "Apposcopy: Semantics-based detection of Android malware through static analysis," in *Proceedings of the 22nd ACM SIGSOFT*

International Symposium on Foundations of Software Engineering, Association for Computing Machinery, pp. 576–587, Hong Kong, China, November 2014.

[4] L. Cen, C. S. Gates, L. Si, and N. Li, "A probabilistic discriminative model for android malware detection with decompiled source code," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 4, pp. 400–412, 2015.

[5] L. Onwuzurike, E. Mariconti, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "MaMaDroid: Detecting android malware by building Markov chains of behavioral models (extended version)," *ACM Trans. Priv. Secur*, vol. 22, no. 2, p. 34, 2019.

[6] K. Tian, D. Yao, B. G. Ryder, G. Tan, and G. Peng, "Detection of repackaged android malware with code-heterogeneity features," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 1, pp. 64–77, 2020.

[7] D. Kong and G. Yan, "Discriminant malware distance learning on structuralinformation for automated malware classification," *ACM SIGMETRICS - Performance Evaluation Review*, vol. 41, no. 1, pp. 347-348, 2013.

[8] B. Yu, Y. Fang, Q. Yang, Y. Tang, and L. Liu, "A survey of malware behavior description and analysis," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 5, pp. 583–603, 2018.

[9] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and efficient behavior-based android malware detection and prevention," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 1, pp. 83–97, 2018.

[10] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: Android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.

[11] M. Zheng, M. Sun, and J. C. S. Lui, "DroidTrace: A ptrace based Android dynamic analysis system with forward execution capability," in *Proceedings of the 2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 128–133, Nicosia, Cyprus, August 2014.

[12] O. Somarriba, U. Zurutuza, R. Uribeetxeberria, L. Delosières, and S. Nadjm-Tehrani, "Detection and visualization of android malware behavior," *JECE*, vol. 2016, p. 6, 2016.

[13] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security, Association for Computing Machinery*, p. 4, Pittsburgh, Pennsylvania, USA, JULY 2011.

[14] N. McLaughlin, J. M. d. Rincon, B. Kang et al., "Deep android malware detection," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, Association for Computing Machinery*, pp. 301–308, Scottsdale, Arizona, USA, 2017.

[15] A. Kumar, K. P. Sagar, K. S. Kuppusamy, and G. Aghila, "Machine learning based malware classification for Android applications using multimodal image representations," in *Proceedings of the 2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pp. 1–6, Coimbatore, India, January 2016.

[16] F. M. Darus, S. N. A. Ahmad, and A. F. M. Ariffin, "Android malware detection using machine learning on image patterns," in *Proceedings of the Cyber Resilience Conference*, Putrajaya, Malaysia, November 2018.

[17] N. P. Poonguzhali, T. Rajakamalam, S. Uma, and R. Manju, "Identification of malware using CNN and bio-inspired technique," in *Proceedings of the 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, pp. 1–5, Pondicherry, India, March 2019.

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[19] J. Gu, Z. Wang, J. Kuen et al., "Recent advances in convolutional neural networks," *Computer Science*, vol. 1, 2015.

[20] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, Honolulu, HI, USA, July 2017.

[21] S. Zia, B. Yüksel, D. Yüret, and Y. Yemez, "RGB-D object recognition using deep convolutional neural networks," in *Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 887–894, Venice, Italy, October 2017.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Salt Lake City, UT, USA, June 2018.

[23] T. H. Huang and H. Kao, "R2-D2: ColoR-inspired convolutional NeuRal network (CNN)-based AndroiD malware detections," in *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*, pp. 2633–2642, Seattle, WA, USA, December 2018.

[24] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, June 2018.

[25] H. Divandari, B. Pechaz, and M. Vafaie, "Malware detection using Markov blanket based on opcode SequencesMalware detection using Markov blanket based on opcode sequences," in *Proceedings of the 2015 International Congress on Technology, Communication and Knowledge (ICTCK)*, Mashhad, Iran, November 2016.