Article

A Multifaceted Deep Generative Adversarial Networks Model for Mobile Malware Detection

Fahad Mazaed Alotaibi [1,†] **and Fawad** [2,*,†]

[1] Department of Information Systems, Faculty of Computing and Information Technology (FCIT), King Abdulaziz University, Jeddah 22254, Saudi Arabia
[2] College of Dentistry, Chosun University, Gwangju 61452, Korea
* Correspondence: fawadmsee20@gmail.com
† These authors contributed equally to this work.

Abstract: Malware's structural transformation to withstand the detection frameworks encourages hackers to steal the public's confidential content. Researchers are developing a protective shield against the intrusion of malicious malware in mobile devices. The deep learning-based android malware detection frameworks have ensured public safety; however, their dependency on diverse training samples has constrained their utilization. The handcrafted malware detection mechanisms have achieved remarkable performance, but their computational overheads are a major hurdle in their utilization. In this work, Multifaceted Deep Generative Adversarial Networks Model (MDGAN) has been developed to detect malware in mobile devices. The hybrid GoogleNet and LSTM features of the grayscale and API sequence have been processed in a pixel-by-pixel pattern through conditional GAN for the robust representation of APK files. The generator produces syntactic malicious features for differentiation in the discriminator network. Experimental validation on the combined AndroZoo and Drebin database has shown 96.2% classification accuracy and a 94.7% F-score, which remain superior to the recently reported frameworks.

Keywords: classification; generative adversarial networks; malware detection

check for updates

Citation: Mazaed Alotaibi, F.; Fawad. A Multifaceted Deep Generative Adversarial Networks Model for Mobile Malware Detection. *Appl. Sci.* **2022**, *12*, 9403. https://doi.org/10.3390/app12199403

Academic Editor: Amalia Miliou

Received: 31 August 2022
Accepted: 15 September 2022
Published: 20 September 2022

1. Introduction

The mobile device has enormously penetrated society in the last few years as a result of enriched interactive features and cost-effectiveness. The advancement of sophisticated sensing devices has brought an exceptional boost to its employment in routine activities. The software developers have made efforts to develop multiple application software related to the academics, health, security, and entertainment sectors. A huge bulk of such application software is publicly available online to assist the end-users. Moreover, online available applications are freely downloaded and installed on user devices without spending any money. The most common operating systems for mobile devices are the Android and iOS operating systems. The software applications interoperable with these operating systems are available on the Google play store and also on third-party software platforms. The Android OS is the primary target for intruders to steal money and the privacy of the public. Malware is malicious software that attacks the operating system of the mobile and attacks the privacy of the users by stealing the credentials of the user. Malware software mostly attacks the android operating system whenever the user unknowingly installs unlicensed apps. The intrusion of malicious software variants through spyware, spoofing, hacking, and phishing has developed an alarming security thread for public credentials and private data. The phishing attacks deceive the recipient by persuading him to click the malicious URL leading to the installation of malware. The open-source code available contains malicious codes including trojan horses, riskware, spyware, adware, and ransomware [1,2]. The malware variants increase their production due to the use of intelligent malware-producing agents such as Zeus, SpyEyea and Dos [3,4]. The privacy and security of the public have recently been researched to develop the privacy-aware applications [5], permissions-assisting application [6], and the malware detection frameworks [7,8]. Anti-virus apps have been developed such as lookout, Norton, and Coodo mobile security, depending on signatures for the detection of malware. The malware signatures are the random snippets and binary patterns extracted from the data samples. The anti-virus developers use the cryptographic signature hash data. The signature data are stored initially to categorize the input data by providing the application [7]. However, malware can encounter the signatures-based detection mechanism by varying a subset of its software sections with preserved semantics.

Various researchers have developed machine learning methods to detect malware in the Android operating system of mobile devices. The developed frameworks can broadly be classified into static and dynamic analysis methods. The static models decompile the application's codes in the installed package data. Moreover, the static features including the application program [9] interface, permission sequence [10], and metadata [11] are extracted in the recompilation process. The model is trained concerning the extracted set of static features for the robust classification of malicious activity by malware. The dynamic analysis models employ the reliable data of the applications, including the system calls [12] and traffic traces [13]. The malware identification model employs a dynamic set of features for the training and classification stage. The dimension of the static and dynamic feature set is extensive, and the model suffers from dimensionality issues. The details of features and samples of the OmniDroid dataset [14] are presented in Table 1. The irrelevant features increase the complexity of the malware detection framework; therefore, preprocessing methodologies have been developed to reduce the feature dimension by removing the unwanted feature values. However, the unsupervised clustering-based malware detection methods that have been developed for detecting zero-day type malware cannot adopt the feature reduction technique to reduce the complexity and dimensionality of the framework [15].

The supervised model developed based on a handcrafted method with a feature reduction mechanism has resulted in a less complex malware identification system. The supervised method consisting of both handcrafted and deep learning-based approaches has been developed by researchers to decrease the complexity while preserving the performance of the model. Various deep learning methods require the conversion of malware into an image in the preprocessing stage followed by the feature extraction and classification stages. The deep learning methods have not only improved the classification performance but also reduced effort for the identification of robust features in the overall system. However, the deep learning model requires a huge set of examples for efficient training. In practice, it is not possible due to limited set of malware samples available. Therefore, the deep learning frameworks are bounded to achieve a constrained accuracy. To withstand the hurdle of limited data samples, various researchers have employed the oversampling phenomena by creating the syntactic samples in SMOTE [16]. The limitation of such methods is that mostly, the re-using of existing samples is adopted instead of creating new samples.

Generative Adversarial Networks (GAN) [17] developed consisting of generator and discriminator neural networks produce the adversarial samples to extend the dataset, leading to false classification results. The generator part produces a fake set of samples, while the discriminator part differentes between the fake and real samples. The direct supply of real samples to the GAN leads to false classification results; therefore, a preprocessed sample is required to increase the invariance and robustness in the overall framework. To increase the robustness and invariance of the classification model, a Multifaceted Deep Generative Adversarial Networks Model (MDGAN) has been developed in this work. Various GAN variants exist in the literature which includes the deep convolutional generative adversarial networks (DCGAN) [18], which was developed for unsupervised learning consisting of multiple architectural constraints. The SinGAN [19,20] is developed to learn the image patch distribution at various scales for recognition based on a single train image. The models require RGB input images and mostly focus on the input data type; however;

the proposed MDGAN employs the multi-face input consisting of the 2D grayscale image features concatenated to the LSTM binary sequence features set for the detection of malware types.

Table 1. OmniDroid [14] dataset features and samples.

Feature Type	Dimension	Malware	Benign
API	2128	11,000	11,000
Permission-API	7629	11,000	11,000
FlowDroid-API	3089	11,000	11,000
Dynamic-API	5932	11,000	11,000

The remaining paper is organized as follows. Section 2 includes the literature review portion that introduces the related works. The proposed methodology portion includes the details of the proposed MDGAN framework. The results section presents the performance of the proposed model in comparison to recently reported works.

2. Literature Review

The android operating system in mobile devices is vulnerable to critical issues including malware attacks. The public privacy and security issues lead to an extreme financial loss of the vendor's capital expenses [21]. The earliest research on malware detection can broadly be classified into two classes: non-machine learning and machine learning-based methods. The non-machine learning frameworks rely on the signature data that consists of static and dynamic analysis approaches. The static method extracts the features from the static code files, whereas the dynamic analysis employs the features of the executing code. The non-machine learning-based method for malware identification is time-consuming and depends solely on the developer's expertise [22]. Moreover, non-machine learning-based methods also required specialized software environments and computational resources [23]. In comparison, the machine learning methods are more precise and less complex depending on the feature extraction model and the classification model. The feature extraction from the input data being broadly classifier into handcrafted and the deep learning methods considers the patterns of the input value concerning the neighbors, whereas the classification model utilizes the pre-trained feature with an associated category label to classify the test data. Various classifiers such as Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbors, and Decision Tree are utilized for the categorization of data into malware and non-malware files. To enhance the classification performance, the malware is first converted into an image before the feature extraction and classification stage. The machine learning methods are based on both handcrafted feature models and deep-leaning based feature extraction models. The image classification task requires a robust and distinctive feature set that remains invariant in the presence of variation in geometry and photometry. Malware identification requires a robust set of features and a stable categorization framework. The conversion of malware into the image type RGB [24], grayscale [25] and binary, consisting of texture, shape, contour, and color distribution can be represented both with handcrafted and DNN methods. The objects within the image are described with the help of color, texture, and shape information [26]. However, due to intra-class variations in color, shape, and illuminations, the image representation becomes quite challenging through simple statistical information. Therefore, stable feature values are identified in the image that do not change when the orientation, scale, and illumination of the objects in the image change. The handcrafted approach for the features extraction process consists of the texture descriptors such as Local Binary Pattern (LBP) [27], Histogram of Oriented Gradient (HoG) [28], and Gray Level Co-occurrence Matrix (GLCM) [29]. The main aim of choosing LBP, HoG, and GLCM is their robustness to noise and invariance to changes in scale, orientation, and the illumination of the objects in the input image. The concatenated shape of all three features representing the same image as a whole is used for the recognition of the objects in the input image. The input image is represented with a multiple features set to bring further distinctiveness and stability to the classification process of the framework. The limitation of such methods is that the handcrafted feature used in this method requires expert knowledge for the extraction of a robust feature set. Furthermore, the feature set designed for one dataset does not work perfectly on the new data files where the new samples of malware are introduced. The DNN and handcrafted model developed so far require a huge set of training samples and require expensive hardware to categorize data precisely. The research aims to develop a highly accurate least complex algorithm for the identification of malware files, which suffers from the constraint on the training samples.

The features extracted from the 2D grayscale visualization of the binary data can help in differentiating various regions of the data file. The features include texture and intensity-based information which cannot be extracted in the binary sequence data. Although a disparity exists between the malware images and the real natural image, the transfer learning through GoogleNet has created a structure that results in a robust set of generic feature values that enhance the classification performance. Therefore, we proposed to combine the features obtained with the image and data sequence to bring invariance and robustness to the malware classification procedure. Various types of neural networks have been developed in the literature for the classification of the ImageNet dataset. The DNN model varies in depth, layer configuration, and size. Some very famous DNN models include AlexNet, ResNet, DagNet, Vgg-16, inceptionV3, GoogleNet, and Yolo models [30]. The models can classify the images very precisely but require a huge set of training data, which is not available in the case of malware data. Moreover, the DNN models also suffer from the issue of uncertainty when they receive out-of-domain input. In [31], the framework has been developed to detect the out-of-distribution samples to resolve the issue of uncertainty. In [32], GAN is used for malware data augmentation, and the imbalance set of the sample has been normalized by generating a training sample from the original set of fewer data. In [33], a vision-based multi-classification approach is developed for IoT malware detection. In [34], the forest penalizing attribute-based malware detection method is developed to classify APK files into malware and non-malware data. The data sample constraints are resolved by introducing the Generative Adversarial Network (GAN), which can generate synthetic samples with the discriminator capable of differentiating the synthetic and real samples. GANs were developed to improve the learning mechanism of the deep neural network by the adversarial learning technique. The parallelization property of the generating of synthetic samples makes the GANS superior to the simple generative algorithms such as PixelCNN [35] and FVBNs [36]. In [37], the framework Malware Generative Adversarial Network (MalGAN) is introduced to generate adversarial examples to attack for malware identification in the data. MalGAN, consisting of the generator part and the discriminator part, remains flexible and defensive in malware file recognition in the data [38].

3. Proposed Methodology

The proposed framework given in Figure 1 shows the overall architecture of the Multifaceted Deep Generative Adversarial Networks Model (MDGAN), which depends on multiple representations at the input face. The proposed framework consists of three main sections to detect the malware in the APK files. The input APK package consisting of various scripts is initially preprocessed and transformed into a binary image and the API sequence file. In the second phase, the binary image is subjected to GoogleNet to extract the distinctive feature part. The API sequence has also proceeded through the LSTM network for the stable set of features. The features are concatenated to obtain a single multi-face vector representation of the APK script files. In the third phase, the GANs are used to extend the trained data through its generator network and then discriminate and classify the input test multi-face feature into malware and non-malware classes.

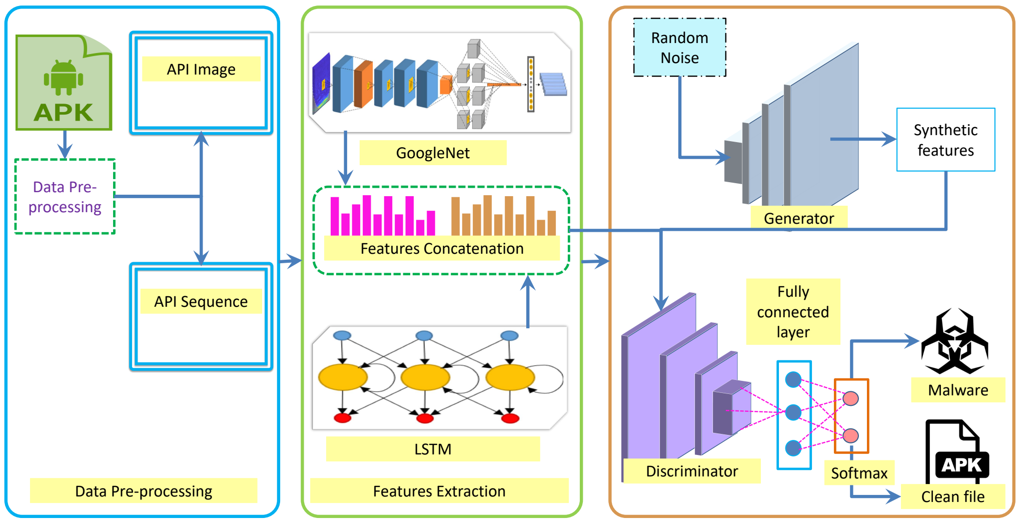
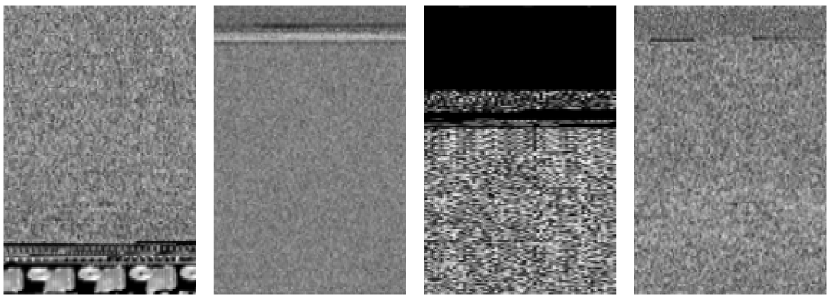


Figure 1. Proposed Framework.

3.1. Data Preprocessing

The input APK file is preprocessed and the malware binary is converted into a grayscale image. A few sample grayscale images of the malware family have been presented in Figure 2. Moreover, to enhance the classification performance, the input APK file is converted to an API sequence for 1D DNN feature extraction. The malware binary is loaded into a 1D array with an 8-bit binary stream transformed into a 2D grayscale image. The input android APK file consists of Manifest.xml for app description, Classes.dex for executable functions, Res directory to store data files, Lib directory for code storage, META-INF for app certificates, Resources.arsc for compiled resources, and Assets directory for Maintainance and upgrade. In the data preprocessing stage, the executable binary files are transformed into 8-bit un-signed segments that are then transformed into 2D grayscale image pixel values, as shown in Figure 3. The width and height of the input 2D image are kept as 224 × 224, which is compatible with the input layer of the GoogleNet DNN model.



(**a**) DroidDream (**b**) FakeDoc (**c**) GinMaster (**d**) DroidKungFu

Figure 2. Sample grayscale images from the malware family.

The API sequence is achieved in two stages. Initially, the word vocabulary vector is created from the API file; then, the API word vector is transformed into an API sequence consisting of numeric integer values. The API execution sequence is analyzed to identify the number of unique words, and then, the numeric value is assigned to each word of the vocabulary, as shown in Figure 3.

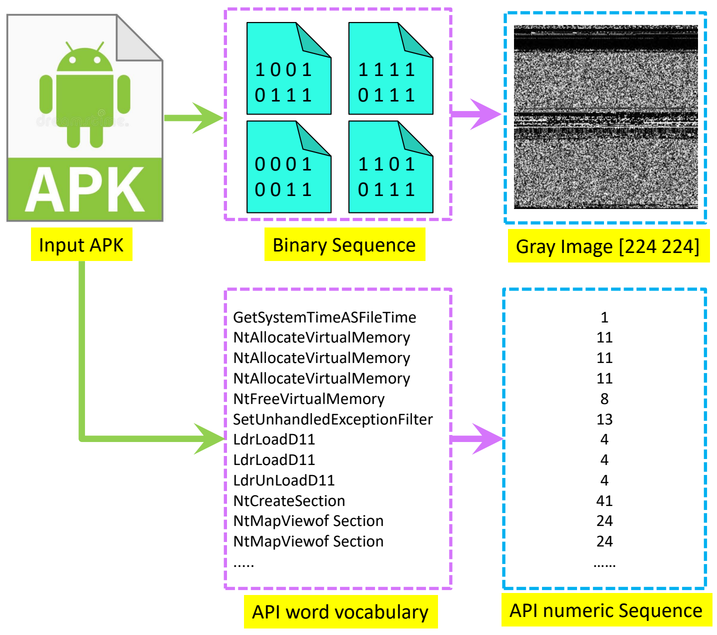


Figure 3. Data Preprocessing.

3.2. Raw Feature Extraction

The raw feature values have been obtained from the multi-face image and sequential numeric data representation of the input APK files. The input image file is obtained through data preprocessing and has been reshaped into a 224 $\times$ 224 image file to become compatible with the input layer of the GoogleNet. The GoogleNet is a 22-layer deep neural network model, consisting of various convolutional layers, pooling layers, inception layers, and fully connected layers employed for the extraction of feature values. The pre-trained GoogleNet is applied through the transfer learning method on the APK image to represent the APK file. The raw feature not only consists of the GoogleNet fully-connected layer variables but also LSTM output feature values extracted from the API sequence. The dimension of the GoogleNet feature vector is 1000 variables, while the dimension of the LSTM feature vector is 64 variables. Both GoogleNet and the LSTM feature values collectively create feature vectors of 1064 variable feature vectors.

3.3. Generative Adversarial Network

The implicit GAN model reported in [17] failed to estimate the probability density of the randomly sampled input data. With the complexity of sampling in multi-dimensional data, the GAN failed to generate the synthetic sample for a subset of data. The contingent GAN model resolved the issue by including the conditional constrained χ on the input abstraction layer of the generator network and its discriminator part.

$$G_{min}D_{max}f(G,D) = \zeta_{r\sim p(r)}[logD(r|\chi)] + \zeta_{\tau\sim p(\tau)}[log(1-D(G(r|\chi))] \quad (1)$$

The mapping $f : \{\tau, \chi\} \to r$ transforms the GAN from an unsupervised to a partially supervised network. The objective function in Equation (1) defines the conditional GANs (CGAN), with the real sample denoted by variable r and the synthetic by variable χ. The f denotes the CGAN, with $p(r)$ as the probability of the real sample, and $p(\tau)$ denotes the probability of the random noisy input sample. The symbol ζ with subscript $r \sim p(r)$ denotes the expected value of the random real samples r, while $\zeta_{\tau\sim p(\tau)}$ denotes the expected value of the synthetic samples τ. The variables G and D represent the generator and the discriminator, respectively.

In the conditional GANs, the generator part produces the synthetic samples as much similar to the original content, whereas the discriminator part has to differentiate in the original and synthetic samples. The issue of pattern collapse has been resolved by employing the pixel-to-pixel sample selection in the conditional GANs. The objective function of the pixel-to-pixel conditional GAN g is given in Equation (2).

$$g(G, D) = \zeta_{r,\chi}[logD(r|\chi)] + \zeta_{r,\tau}[log(1 - D(x, G(r|\tau)))] \tag{2}$$

where the symbol g denotes the pix2pix conditional GAN model. The loss function of the proposed multi-face GANs consisting of pixel-to-pixel conditional GAN depending on the hybrid raw feature is represented with Equations (3) and (4).

$$L(g) = \zeta_{v,r,\chi}[||\chi - g(v, \tau)||] \tag{3}$$

$$L^* = argG_{min}D_{max}L_g(G, D) + \lambda L(g) \tag{4}$$

whereas the symbol v denotes the input raw feature vector. The variable λ works as an adjustment parameter in Equation (4); when the λ is zero, then the loss function of the pix2pix CGAN turns to a conditional GAN model.

The proposed multi-face contingent pixel-to-pixel version of the GAN model is presented in Figure 4.

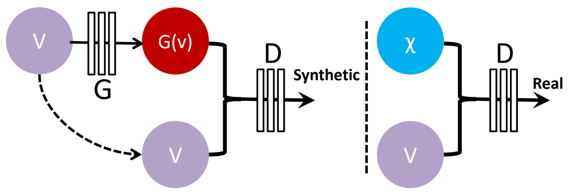


Figure 4. Multi-face GAN.

The pixel-to-pixel considers the feature values individually in the raw feature set. The raw feature is divided into N $\times$ 1 and patch, and the discriminator judges the patch instead of the whole feature vector for authenticity. The N dimension equals 128, which provided a higher recognition in the proposed model. The classification method of LeNet-5 [39] is employed, and the features extracted are classified with the fully connected layer to reduce the complexity of the computational procedures. The network parameters are trained separately to optimize the classification performance of the network.

4. Experimental Results

4.1. Datasets

The dataset employed in the experimental validation of the proposed model consists of 5546 malicious and 5831 benign Android APK files. The benign files were obtained from the AndroZoo database, where the (.CSV) file annotates the description of each app file in the dataset. The malware files collected from the Derbin database consisted of 179 malware families collectively. The malware family files such as Fake-Installer, Droid-KungFu, Plankton, Op-fake, Base-Bridge, Gin-Master, Iconosys, Kmin, Adrd, Geinimi, Fake-Doc, and DroidDream are considered in the database. The details of the dataset family samples have been shown in Figure 5. The complete database is divided with a ratio of 7:3 for training and test sets, respectively.

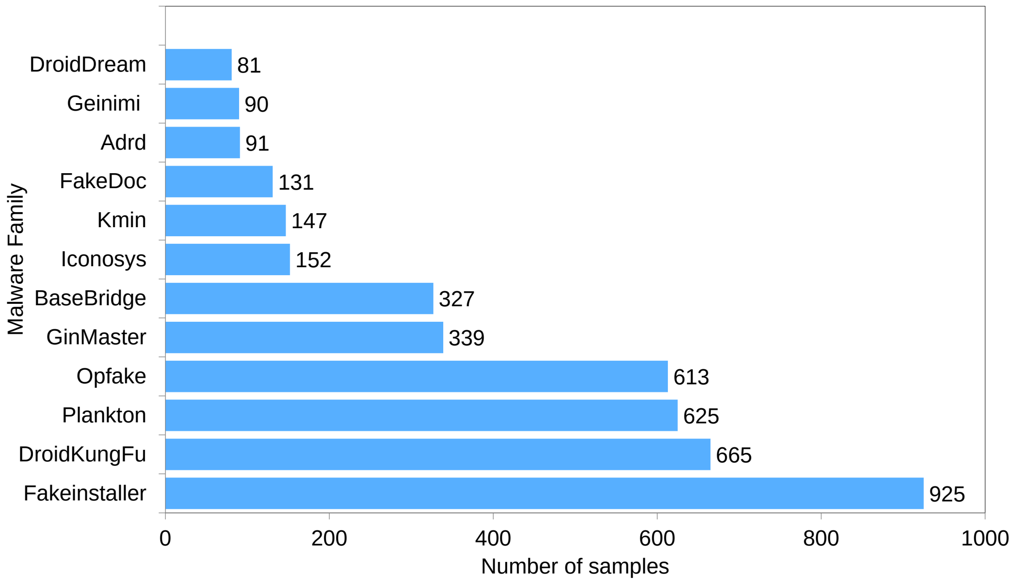


Figure 5. Variants of the Malware family in the database.

4.2. Evaluation

The performance evaluation of the proposed framework is shown based on the confusion matrix given in Figure 6. The horizontal axis displays the actual category index, while the vertical axis represents the predicted labels. The list of overall evaluation parameters has been displayed in Table 2. The $T_{i,j}$ in the table denotes the distribution of actual ith values in the jth prediction class of the nth family.

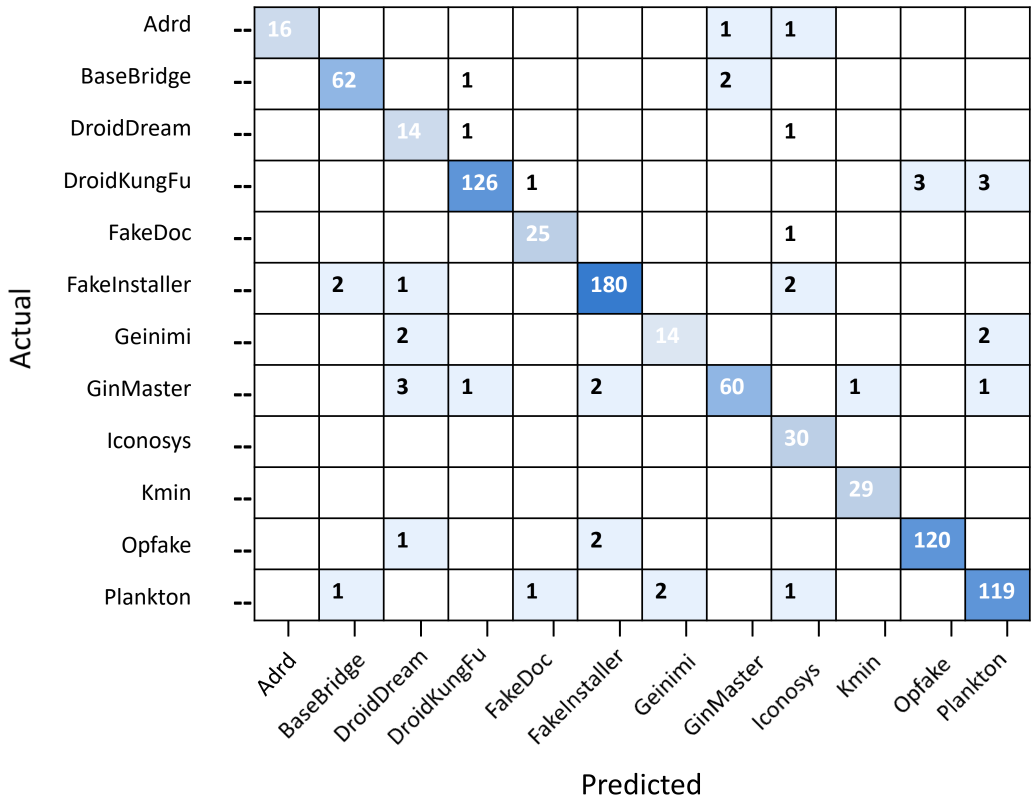


Figure 6. Proposed model classification performance measurement through confusion matrix.

Table 2. Evaluation Parameters.

Parameter	Expression	
Mean Precision	$\frac{1}{n}\sum_{i=1}^{n}\frac{T_{ii}}{\sum_{j=1}^{n}T_{ji}}$	(5)
Mean Recall	$\frac{1}{n}\sum_{i=1}^{n}\frac{T_{ii}}{\sum_{j=1}^{n}T_{ij}}$	(6)
Mean Fscore	$\frac{1}{n}\sum_{i=1}^{n}\frac{2 * Precision_i * Recall_i}{Precision_i + Recall_i}$	(7)

The overall accuracy obtained with the expression is given in Equation (8), where the parameters T_P, T_N, F_P, and F_N denote the true positive, true negative, false positive, and false negative respectively.

$$Overall\text{-}Accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \tag{8}$$

4.3. Classification Results

The generator networks generate adversarial synthetic samples similar to malicious samples when the conditional synthetic χ and the real $\times$ match. This enhances the training data for the machine learning procedure used by the proposed MDGAN model. The confusion matrix shown in Figure 6 displays the effectiveness of the adversarial training in the proposed MDGAN method combined with the multi-face data input mechanism that has learned and categorized the malware family. The malware family samples count shown in Figure 5 displays the total number of samples present in each malware type of the family. The malware family classification result shown in the confusion matrix witnesses the superiority of the proposed framework over other recently reported mechanisms. The DroidDream and Iconosys have not learned the feature set more effectively and obtained 85.7% and 88.5% precision values with their 81 and 152 samples in the family as given in Table 3. The Adrd and Geinimi both have the highest inter-class variation due to which they have achieved the highest precision of 100%, while the FakeInstaller achieved the second highest precision of 98.9% due to its highest samples strength of 925 samples in the family. The mean precision value obtained by the proposed MDGAN is 95.1%, which is higher than DNN-RNN, CNN-raw opcodes, DroidDetective, API calls-yerima, and CNN-BiLSTM-NB with precision of 90%, 87.2%, 89.5%, 94.3%, and 90.0%, respectively. The mean recall value of MDGAN is 94.6%, which is slightly lower when compared to the 96% value attained by DroidDetective given in Table 4. The mean F1-score of MDGAN is 94.7%, which is superior to DNN-RNN, CNN-raw opcodes, DroidDetective, API calls-yerima, and CNN-BiLSTM-NB with F1-score values of 87.1%, 86.2%, 92.1%, 92.3%, and 86.3% respectively. The mean accuracy achieved by MDGAN is 96.2%, which is superior to DNN-RNN, CNN-raw opcodes, DroidDetective, API calls-yerima, and CNN-BiLSTM-NB with a mean accuracy of 90%, 87.4%, 86.0%, 91.8%, and 88.1%.

Table 3. Confusion matrix summarization.

	Train			Test		
Family	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
Adrd	1	0.892	0.951	1	0.889	0.941
BaseBridge	0.941	0.964	0.955	0.939	0.953	0.945
DroidDream	0.872	0.912	0.882	0.857	0.909	0.882
DroidKungFu	0.979	0.952	0.962	0.962	0.948	0.954
FakeDoc	0.981	1	0.991	0.961	1	0.980
FakeInstaller	0.999	0.982	0.986	0.989	0.973	0.980
Geinimi	1	0.871	0.933	1	0.866	0.928
GinMaster	0.921	0.911	0.914	0.909	0.909	0.909
Iconosys	0.892	1	0.944	0.885	1	0.939
Kmin	0.989	0.973	0.982	0.983	0.968	0.975
Opfake	0.988	0.972	0.981	0.976	0.969	0.972
Plankton	0.970	0.979	0.970	0.952	0.975	0.963
Average	0.961	0.951	0.954	0.951	0.946	0.947
Accuracy			**0.973**			**0.962**

Table 4. Comparison with state-of-the-art methods.

		Train				Test			
Ref.	**Methodology**	**Precision**	**Recall**	**F-Measure**	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**Accuracy**
[40]	DNN/RNN	90.4%	84.9%	87.5%	90.7%	90.0%	84.0%	87.1%	90.0%
[40]	CNN-raw opcodes	88.3%	85.8%	86.5%	87.9%	87.2%	85.5%	86.2%	87.4%
[41]	DroidDetective	89.8%	96.4%	92.5%	96.1%	89.5%	96.0%	92.1%	96.0%
[42]	API calls, Yerima	94.6%	91.9%	92.7%	91.9%	94.3%	91.7%	92.3%	91.8%
[43]	CNN–BiLSTM-NB	90.5%	87.4%	86.8%	88.6%	90.0%	87.1%	86.3%	88.1%
Ours	**MDGAN**	**95.9%**	**94.9%**	**94.8%**	**96.5%**	**95.1%**	**94.6%**	**94.7%**	**96.2%**

5. Conclusions

The Multifaceted Deep Generative Adversarial Networks Model (MDGAN) was developed to identify malware APIs installed on mobile devices. The android devices operate the open access applications available on the Google play store, which contain malware files that affect the APK file by performing malicious activities to leak the privacy and safety of the users. The proposed framework is multi-face with hybrid DNN API-image and API sequence features, and it is interfaced with conditional GAN operating on the pixel-to-pixel sample selection. The proposed MDGAN achieved superior performance compared to the existing works with 95.1%, 94.6%, 94.7%, and 96.2% mean precision, recall, F1-score, and average accuracy, respectively.

Author Contributions: F.A.: investigation and project administration, and funding acquisition, F.: software, validation, writing, and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research work was funded by the institutional fund projects under grant no. (IFPRP: 290-611-1442). Therefore, the authors gratefully acknowledge technical and financial support from the Ministry of Education and King AbdulAziz University, DSR, Jeddah, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All suported data included in the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Aboaoja, F.A.; Zainal, A.; Ghaleb, F.A.; Al-rimy, B.A.S.; Eisa, T.A.E.; Elnour, A.A.H. Malware Detection Issues, Challenges, and Future Directions: A Survey. *Appl. Sci.* **2022**, *12*, 8482. [CrossRef]
2. Chen, D.; Wawrzynski, P.; Lv, Z. Cyber security in smart cities: A review of deep learning-based applications and case studies. *Sustain. Cities Soc.* **2021**, *66*, 102655. [CrossRef]
3. Awan, M.J.; Farooq, U.; Babar, H.M.A.; Yasin, A.; Nobanee, H.; Hussain, M.; Hakeem, O.; Zain, A.M. Real-time DDoS attack detection system using big data approach. *Sustainability* **2021**, *13*, 10743. [CrossRef]
4. Ferooz, F.; Hassan, M.T.; Awan, M.J.; Nobanee, H.; Kamal, M.; Yasin, A.; Zain, A.M. Suicide bomb attack identification and analytics through data mining techniques. *Electronics* **2021**, *10*, 2398. [CrossRef]
5. Perera, C.; Barhamgi, M.; Bandara, A.K.; Ajmal, M.; Price, B.; Nuseibeh, B. Designing privacy-aware internet of things applications. *Inf. Sci.* **2020**, *512*, 238–257. [CrossRef]
6. Azad, M.A.; Arshad, J.; Akmal, S.M.A.; Riaz, F.; Abdullah, S.; Imran, M.; Ahmad, F. A first look at privacy analysis of COVID-19 contact-tracing mobile applications. *IEEE Internet Things J.* **2020**, *8*, 15796–15806. [CrossRef]
7. Tam, K.; Feizollah, A.; Anuar, N.B.; Salleh, R.; Cavallaro, L. The evolution of android malware and android analysis techniques. *ACM Comput. Surv.* **2017**, *49*, 1–41. [CrossRef]
8. Zheng, M.; Sun, M.; Lui, J.C.S. Droid Analytics: A signature based analytic system to collect, extract, analyze and associate android malware. In Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Melbourne, Australia, 16–18 July 2013; pp. 163–171.
9. Seo, S.H.; Gupta, A.; Sallam, A.M.; Bertino, E.; Yim, K. Detecting mobile malware threats to homeland security through static analysis. *J. Netw. Comput. Appl.* **2014**, *38*, 43–53. [CrossRef]
10. Sharma, K.; Gupta, B.B. Mitigation and risk factor analysis of android applications. *Comput. Electr. Eng.* **2018**, *71*, 416–430. [CrossRef]
11. Potharaju, R.; Newell, A.; Nita-Rotaru, C.; Zhang, X. Plagiarizing smartphone applications: Attack strategies and defense techniques. *ACM Int. Symp. Eng. Secure Softw. Syst.* **2012**, *7159*, 106–120.
12. Xiao, X.; Xiao, X.; Jiang, Y.; Liu, X.; Ye, R. Identifying Android malware with system call co-occurrence matrices. *Trans. Emerg. Telecommun. Technol.* **2018**, *27*, 675–684. [CrossRef]
13. Chen, Z.; Yan, Q.; Han, H.; Wang, S.; Peng, L.; Wang, L.; Yang, B. Machine learning based mobile malware detection using highly imbalanced network traffic. *Inform. Sci.* **2018**, *433*, 346–364. [CrossRef]
14. Martin, A.; Lara-Cabrera, R.; Camacho, D. Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset. *Inf. Fusion* **2019**, *52*, 128–142. [CrossRef]
15. Pai, S.; Troia, F.D.; Visaggio, C.A.; Austin, T.H.; Stamp, M. Clustering for malware classification. *J. Comput. Virol. Hacking Tech.* **2018**, *13*, 95–107. [CrossRef]
16. Chawla V, N.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
17. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Bengio, Y. Generative adversarial nets. *Adv. Neural Inf. Process. Syst.* **2020**, *63*, 139–144.
18. Radford, A.; Metz, L.; Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv* **2015**, arXiv:1511.06434.
19. Shaham, T.R.; Dekel, T.; Michaeli, T. Singan: Learning a generative model from a single natural image. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 4570–4580.
20. Akhenia, P.; Bhavsar, K.; Panchal, J.; Vakharia, V. Fault severity classification of ball bearing using SinGAN and deep convolutional neural network. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2022**, *236*, 3864–3877. [CrossRef]
21. Hammad, B.T.; Jamil, N.; Ahmed, I.T.; Zain, Z.M.; Basheer, S. Robust Malware Family Classification Using Effective Features and Classifiers. *Appl. Sci.* **2022**, *12*, 7877. [CrossRef]
22. Aslan, O.A.; Samet, R. A comprehensive review on malware detection approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
23. Wan, Y.L.; Chang, J.C.; Chen, R.J.; Wang, S.J. Feature-selection-based ransomware detection with machine learning of data analysis. In Proceedings of the 2018 3rd International Conference on Computer and Communication Systems (ICCCS), Nagoya, Japan, 27–30 April 2018; pp. 85–88.
24. Zhang, Y.; Yang, Y.; Wang, X. A Novel Android Malware Detection Approach Based on Convolutional Neural Network. In Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, Guiyang, China, 16–18 March 2018; pp. 144–149.
25. Jung, J.; Choi, J.; Cho, S.J.; Han, S.; Park, M.; Hwang, Y. Android malware detection using convolutional neural networks and data section images. In Proceedings of the RACS '18, Honolulu, HI, USA, 9–12 October 2018; pp. 149–153.
26. Hu, H.; Yang, W.; Xia, G.S.; Lui, G. A color-texture-structure descriptor for high-resolution satellite image classification. *Remote Sens.* **2016**, *8*, 259.
27. Song, T.; Feng, J.; Luo, L.; Gao, C.; Li, H. Robust texture description using local grouped order pattern and non-local binary pattern. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *31*, 189–202. [CrossRef]
28. Patel, C.I.; Labana, D.; Pandya, S.; Modi, K.; Ghayvat, H.; Awais, M. Histogram of oriented gradient-based fusion of features for human action recognition in action video sequences. *Sensors* **2020**, *20*, 7299. [CrossRef] [PubMed]

29. Park, Y.; Guldmann, J.M. Measuring continuous landscape patterns with Gray-Level Co-Occurrence Matrix (GLCM) indices: An alternative to patch metrics? *Ecol. Indic.* **2020**, *109*, 105802. [CrossRef]
30. Agbo-Ajala, O.; Viriri, S. Deep learning approach for facial age classification: A survey of the state-of-the-art. *Artif. Intell. Rev.* **2021**, *54*, 179–213. [CrossRef]
31. Liu, J.Z.; Padhy, S.; Ren, J.; Lin, Z.; Wen, Y.; Jerfel, G.; Lakshminarayanan, B. A Simple Approach to Improve Single-Model Deep Uncertainty via Distance-Awareness. *arXiv* **2022**, arXiv:2205.00403.
32. Chen, Y.M.; Yang, C.H.; Chen, G.C. Using generative adversarial networks for data augmentation in android malware detection. In Proceedings of the 2021 IEEE Conference on Dependable and Secure Computing (DSC), Aizuwakamatsu, Fukushima, Japan, 30 January–2 February 2021; pp. 1–8.
33. Atitallah, S.B.; Driss, M.; Almomani, I. A Novel Detection and Multi-Classification Approach for IoT-Malware Using Random Forest Voting of Fine-Tuning Convolutional Neural Networks. *Sensors* **2022**, *22*, 4302. [CrossRef]
34. Akintola, A.G.; Balogun, A.O.; Capretz, L.F.; Mojeed, H.A.; Basri, S.; Salihu, S.A.; Alanamu, Z.O. Empirical Analysis of Forest Penalizing Attribute and Its Enhanced Variations for Android Malware Detection. *Appl. Sci.* **2022**, *12*, 4664. [CrossRef]
35. Frey, B.J.; Hinton, G.E.; Dayan, P. Does the wake-sleep algorithm produce good density estimators? *Adv. Neural Inf. Process. Syst.* **1996**, *8*, 661–667.
36. Frey, B.J.; Brendan, J.F.; Frey, B.J. *Graphical Models for Machine Learning and Digital Communication*; MIT Press: Cambridge, MA, USA, 1998.
37. Hu, W.; Tan, Y. Generating adversarial malware examples for black-box attacks based on GAN. *arXiv* **2017**, arXiv:1702.05983.
38. Gui, J.; Sun, Z.; Wen, Y.; Tao, D.; Ye, J. A review on generative adversarial networks: Algorithms, theory, and applications. *IEEE Trans. Knowl. Data Eng.* **2021**, *1*, 1–5. [CrossRef]
39. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
40. Mchaughlin, N.; del Rincon, J.M.; Kang, B.; Yerima, S.; Safaei, Y.; Trickel, E.; Zhao, Z.; Doupe, A.; Ahn, G.J. Deep Android Malware Detection. In Proceedings of the ACM on Conference on Data and Application Security and Privacy (CODASPY), Scottsdale, AZ, USA, 2017; pp. 301–308.
41. Liang, S.; Du, X. Permission-combination-based scheme for android mobile malware detection. *IEEE Int. Conf. Commun. (ICC)* **2014**, *1*, 2301–2306.
42. Jerome, Q.; Allix, K.; State, R.; Engel, T. Using opcode-sequences to detect malicious android applications. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, Australia, 10–14 June 2014; pp. 914–919.
43. Zhang, N.; Xue, J.; Ma, Y.; Zhang, R.; Liang, T.; Tan, Y.A. Hybrid sequence-based Android malware detection using natural language processing. *Int. J. Intell. Syst.* **2021**, *36*, 5770–5784. [CrossRef]