6-21-2021

Classification and Analysis of Android Malware Images Using Feature Fusion Technique

Jaiteg Singh
Chitkara University

Deepak Thakur
Chitkara University

Tanya Gera
Chitkara University

Babar Shah
Zayed University, babar.shah@zu.ac.ae

Tamer Abuhmed
Sungkyunkwan University

See next page for additional authors

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Classification and Analysis of Android Malware Images Using Feature Fusion Technique

JAITEG SINGH[1], DEEPAK THAKUR[1], TANYA GERA[1], BABAR SHAH[2], TAMER ABUHMED[3], and FARMAN ALI[4]

[1]Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India
[2]College of Technological Innovation, Zayed University, UAE
[3]Department of Computer Science and Engineering, College of Computing, Sungkyunkwan University, Republic of Korea
[4]Department of Software, Sejong University, Seoul 05006, Korea

Corresponding author: Deepak Thakur (e-mail: deepak.thakur@chitkara.edu.in), Tamer Abuhmed (e-mail: tamer@skku.edu)

"This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C1011198)"

ABSTRACT The super packed functionalities and artificial intelligence (AI)-powered applications have made the Android operating system a big player in the market. Android smartphones have become an integral part of life and users are reliant on their smart devices for making calls, sending text messages, navigation, games, and financial transactions to name a few. This evolution of the smartphone community has opened new horizons for malware developers. As malware variants are growing at a tremendous rate every year, there is an urgent need to combat against stealth malware techniques. This paper proposes a visualization and machine learning-based framework for classifying Android malware. Android malware applications from the DREBIN dataset were converted into grayscale images. In the first phase of the experiment, the proposed framework transforms Android malware into fifteen different image sections and identifies malware files by exploiting handcrafted features associated with Android malware images. The algorithms such as Gray Level Co-occurrence Matrix-based (GLCM), Global Image deScripTors (GIST), and Local Binary Pattern (LBP) are used to extract the handcrafted features from the image sections. The extracted features were further classified using machine learning algorithms like K-Nearest Neighbors, Support Vector Machines, and Random Forests. In the second phase of the experiment, handcrafted features were fused with CNN features to form the feature fusion strategy. The classification performance was evaluated against every malware image file section. The results obtained using the Feature Fusion strategy are compared with handcrafted features results. The experiment results conclude to the fact that Feature Fusion-SVM model is most suited for the identification and classification of Android malware using the certificate and Android Manifest (CR+AM) malware images. It attained an high accuracy of 93.24%.

INDEX TERMS Handcrafted features, Machine learning, malware, Classification, Visualization, Android, Security, Feature fusion

I. INTRODUCTION

Smartphones nowadays are a virtual substitute for any generic computing device. Smartphones have become an integral part of life and users are reliant on their smart devices for making calls, sending text messages, navigation, games, and financial transactions to name a few. This evolution of the smartphone community has opened new horizons for malware developers. There are more than thirty categories available on online app stores like Google Playstore. Among those categories, Games, Business, Lifestyle, Education, Entertainment, and Health & Fitness are found to be the most popular. Users make use of these applications to their maximum advantage and tend to communicate, entertain, business, relax, and educate themselves. The rapid adoption of such applications has resulted in the generation and sharing of sensitive information. Amongst the plethora of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

available mobile operating systems, Android has managed to conquer more than 86% of the total market. Android being a market leader has an open marketplace and a huge community promulgating intensely popular APIs.

The popularity of the Android operating system has also attracted cybercriminals to develop malicious applications to exploit Android users for monetary benefits. The cyber attacks are commonly categorized as Malware, Adware, and Potentially Unwanted Applications (PUA). As per the annual threat report 2020, 57% of the total detected attacks were due to malware. During the COVID-19 pandemic, a notable rise is observed in the number and the severity of cyber-attacks. The finding indicates that 68% of the total reported attacks were related to financial gains. Furthermore, once the malware applications breach into the phone, they can adversely affect the smooth flow of an activity lifecycle paradigm. Activity lifecycle involves various stages such as onCreate(), onStart(), onStop(), and onDestroy() to name a few. These callbacks are important to preserve because they do take care of the normal execution of an Android application. Android-powered devices run the archive file known as Android Package (APK). An APK can be written in renowned languages such as java, C++, and kotlin. The APKs which are of few megabytes (MBs) in size when backed with malicious payloads can harm the user socially, emotionally, and financially. Malware applications tend to hijack the imperative building blocks of the APK known as application components. These components are activities, broadcast receivers, content providers, and services. Malware authors take control of these components and compromise the Android devices by establishing communication with Command and Control (C&C) servers.

Automation and artificial intelligence are on the rise to generate variants of malware families rapidly. Researchers have realized that using signature-based methods, static methods, and dynamic methods are not competing against fast-growing malware variants. Signature-based detection approaches are more prone to code obfuscation and transformation techniques. These approaches also need to keep their database updated every time by appending new malware variants into it. Plenty of time and expertise is invested in manually analyzing the signatures and then extracting them. The static analysis doesn't stand even with trivial transformations [1]–[3]. On the other hand, dynamic analysis is heavy on time and resources [4], [5]. Significant time is required to extract the static and dynamic features for the detection and classification of Android malware. The researchers have proposed various algorithms to build a robust feature set to solve the multiclass problem. Constructing the feature set manually is a tedious task and hence requires more expertise and time. There is a need to deploy better feature reduction techniques or other supplementary techniques to build time-sensitive feature sets in Android malware research.

Visualization-based techniques do not let the application to execute rather it extracts CNN features [6] and handcrafted features for the classification task. Handcrafted features are used to extract the information from the images. These features help to solve classification problems. The algorithms such as GLCM, GIST, and LBP are also known as texture or image descriptors. To perform classification, the aforementioned algorithms must be used in linear combination with machine learning classifiers such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Random Forest (RF). The image-based approaches based on handcrafted features have gained an edge over traditional approaches for malware identification as handcrafted features refer to properties derived using various algorithms using the information present in the image itself. The adopted methodology would study raw bytes of malware code visualized as an image. Such consideration would eventually eliminate the need for decryption, disassembly, reverse engineering, and execution of code. Min-Max Normalization method is considered to investigate the impact of data normalization on the classification performance of malware images. Results produced on normalized and un-normalized are also compared. A total of fifteen unique combinations of the Android malware file structure were used to generate the malware images. This paper is the extension of the work presented by authors in [7]. They have deployed the visualization-based approach infusion with deep learning architecture to classify the Android malware families. We are motivated to improve the classification accuracy by proposing the model based on feature fusion methodology. Feature fusion is constructed by combining the rich features extracted from deep layers of Convolutional Neural Network (CNN) with handcrafted features such as Gray Level Co-occurrence Matrix (GLCM), Global Image deScripTors (GIST), and Local Binary Pattern (LBP).

The manuscript is organized as: section 2 discusses the related work of the study, section 3 lays the foundation for the proposed methodology, section 4 elaborates the results and findings, and section 5 concludes the study.

II. RELATED WORK

Authors in [8] implemented image-based approach to identify the malicious patterns in the code. They mapped the sequence of API pairs to RGB images. After preprocessing and preparation of the data for the neural network, it was fed into the convolutional neural network. They worked on the two-class problem i.e. detecting whether an application is benign or malicious. The authors ran the experiment for 100 epochs with batch size 32. A disassembly process was required to extract the API calls. Authors in [9] consider a dataset of 144 Android permissions. They used the tools such as androguard parser and smali disassembler for the parsing and decompilation process of an APK. Further, they extracted the requested permissions from the disassembled manifest file and mapped it into 12x12 permission vectors as an image. Their dataset contains a total of 2500 Android applications in which 2000 applications were malware samples and 500 applications were benign samples. Further, a

deep learning model was applied to identify and classify the malware samples. Authors in [10], first disassembled the APK file and extracted only the dex bytecode from the file. They converted the dex bytecode into RGB image format and fed it into a convolutional neural network for automatic feature extraction and training. Authors in [11], implemented their Android malware detection approach in two phases. In the first phase, they extracted the dex bytecode from the APK archive and transformed it into RGB images. In the second phase, images were used to train the convolutional neural network. They implemented eight hidden layers in the convolutional neural network and used the softmax function to classify whether the sample is benign or malicious. A better result of precision and recall was observed for malware samples as compared to benign samples. Authors in [12] used the convolutional neural network for detection of Android malware. They had used the Rectified Linear Unit (Relu) activation function as it overcomes the vanishing gradient problem and shows better convergence performance. Furthermore, they had employed the deep autoencoder to reduce the training time by 83% of the convolutional neural network. They had also compared their model with other machine classifiers such as Support Vector Machine (SVM). The accuracy of their proposed model was improved by 5% when compared to the accuracy obtained using the SVM classifier. Authors in [13] implemented multimodal deep learning strategy for Android malware detection. They have used publicly available dataset omnidroid and Knowledge Discovery in Databases (KDD) for training and evaluation of the proposed model. They utilized manual and automatic feature engineering using deep learning architectures. They have used convolutional neural network, deep neural network, and transformer networks to perform feature learning from grayscale images which are generated from dex bytecode, static features i.e. intents and permissions, and dynamic features i.e. system calls respectively. Authors in [14] utilizes the GIST features for the classification of malware families. The classifiers such as Support Vector Machines, K-Nearest Neighbor, Random Forests, and Naive Bayes were used in the experimenation. The results with Support Vector Machines attain the highest accuracy of 92.7%. Authors in [15] transformed the dalvik executable code into two dimensional bytecode matrix. Further, convolutional neural network was used for training and classification task. Convolutional neural network can automatically learn the features from the bytecode files to recognize the malware. Various research areas and trends in Android security domain were studied by the authors using latent semantic analysis technique in [16], [17]. Authors in [18] discusses the alarming challenges in the field of Android security. Authors in [19], implemented the tensorflow models i.e. GoogleNet and ResNet for malware detection. In their work, ResNet proved to be more accurate but consumed a lot of time. Authors in [20] proposed the image texture-based approach to perform the analysis on the code. They combine the image texture features and API calls to train the Deep Belief Network (DBN). DBN is stacked with Restricted Boltzmann Machines (RBN) and Back Propagation (BP). Authors also compared their proposed model with shallow machine learning models such as Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and shallow feed-forward network ANN (Artificial Neural Network) and found that the proposed DBN model was more accurate. Table 1 comprises of a summary of related literature.

The literature survey revealed that approaches of analyzing malware include static analysis and dynamic analysis or maybe the combination of both. The static analysis mainly focuses on disassembling the code, followed by manual investigation to search the malicious patterns in the code. Conversely, dynamic analysis executes the code in the virtual environment and analyzes its execution trace to observe the malicious behavior of an application. The static analysis is helpful in tracing original and full execution paths; therefore, it provides complete code coverage but eventually it suffers from code obfuscation. The sample has to be decrypted first to perform static analysis. The problems of intractable complexity hinder the analysis. Dynamic analysis is more efficient and does not need the executable to be unpacked or decrypted. The suspicious application is monitored in a controlled environment. This process is time and resource consuming. It also raises scalability issues. Moreover, some malicious behavior might be unobserved because the environment does not satisfy the triggering conditions. Furthermore, malware authors make use of automation technology to generate a huge amount of new malware variants, thus posing a big challenge to malware analysts. The present state of art demands the integration of existing primitive techniques with supplementary techniques to achieve an effective solution. Supplementary techniques such as visualization-based analysis should be leveraged to complement the classification of fast-growing Android malware families. It is proven to be effective in determining abnormal modern malicious behavior or security vulnerabilities. Deploying a visualization-based technique, a malware variant can be visualized as an image. An image can capture even small changes. In this paper, the visualization-based technique backed with feature fusion strategy is proposed to reduce the influence of obfuscation by transforming the malware‘s non-intuitive features into fingerprint images followed by the classification of Android malware families. The following section explains the adopted methodology and to undertake a case analysis.

III. MATERIALS AND METHODS

We evaluated our experiments over the DREBIN [38] dataset. This dataset has been adopted by many researchers investigating Android malware. The count of samples of malware families in DREBIN dataset is shown in the Figure 1. The subsequent sections discuss the methodology of the proposed work followed by results and findings. A graphical representation of the proposed methodology is

TABLE 1. Summary of the related work

Ref.	GIST	GLCM	LBP	CNN Features	Other features	KNN	RF	SVM	ANN	CNN	Other model/Features used
[21]	No	No	No	No	-	Yes	No	No	No	No	320
[22]	No	Yes	No	No	-	Yes	No	No	No	No	-
[22]	No	Yes	No	No	-	No	No	Yes	No	No	-
[23]	No	No	Yes	No	-	Yes	No	No	No	No	108
[23]	No	No	Yes	No	-	No	No	Yes	No	No	108
[23]	No	No	Yes	No	-	No	No	No	No	Yes	108
[23]	Yes	No	No	No	-	Yes	No	No	No	No	512
[23]	Yes	No	No	No	-	No	No	Yes	No	No	512
[23]	Yes	No	No	No	-	No	No	No	No	Yes	512
[24]	Yes	No	No	No	-	No	No	No	Yes	No	320
[25]	No	No	No	No	Network, System calls, File system access, Binder transactions	No	No	Yes	No	No	-
[26]	No	No	No	No	System calls	No	No	No	No	Yes	-
[27]	No	No	No	No	Discrete Wavelet Transformation	No		Yes	No	No	-
[28]	No	No	No	Yes	-	No	No	No	No	Yes	-
[29]	Yes	No	No	No	Gabor wavelet, Discrete Wavelet Transformation	No	No	Yes	No	No	56
[29]	Yes	No	No	No	Gaabor wavelet, Discrete Wavelet Transformation	Yes	No	No	No	No	56
[30]	No	No	No	No	Principle Componet Analysis (PCA) features	Yes	No	No	No	No	12
[30]	No	No	No	No	Principle Componet Analysis (PCA) features	No	No	Yes	No	No	10
[30]	No	No	No	No	Principle Componet Analysis (PCA) features	No	No	No	Yes	No	52
[31]	No	No	No	Yes	-	No	No	No	No	Yes	-
[32]	No	No	No	No	Gabor based features	No	No	Yes	No	No	512
[32]	No	No	No	No	Intensity and wavelet based	No	No	Yes	No	No	22
[32]	No	No	No	No	Intensity, Wavelet and Gabor based features	No	No	Yes	No	No	534
[33]	Yes	No	No	No	-	Yes	No	No	No	No	512
[33]	Yes	No	No	No	-	No	Yes	No	No	No	512
[34]	Yes	No	No	No	-	Yes	No	No	No	No	512
[34]	No		Yes	No	-	Yes	No	No	No	No	512
[34]	No	Yes	No	No	-	Yes	No	No	No	No	20
[34]	Yes	No	No	No	D-Sift	Yes	No	No	No	No	512
[35]	Yes	No	No	No	-	No	No	Yes	No	No	-
[35]	No	No	No	Yes	-	No	No	No	No	Yes	-
[35]	No	No	No	Yes	-	No	No	No	No	Yes	-
[36]	No	No	No	No	-	No	No	No	No	No	CNN-SVM
[36]	No	No	No	No	-	No	No	No	No	No	GRU-SVM
[36]	No	No	No	No	-	No	No	No	No	No	MLP-SVM
[37]	No	Yes	No	No	-	Yes	No	No	No	No	20
[37]	No	Yes	No	No	-	No	Yes	No	No	No	20

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

illustrated in Figure 2. The following subsections discusses the underlying steps in proposed methodolgy.

A. DESCRIPTION OF FUNDAMENTAL CONCEPTS

With a slight change in malware code, the malware authors can generate numerous malware variants. Any type of malware can be visualized as an image, which can capture even the slightest of the changes. An image has the capability to retain the original semantics of the code. The basic structure of an APK and the process of transforming malware into images are defined in subsequent sections. All files and folders that are contained in the ZIP archive of an APK. These files are binded together to develop an application. The literature survey indicated an inspection of four types of primary resources for identifying malware behaviour [5]. These primary resources include classes.dex, resource, manifest, and certificate files. There is a high propensity that malware developers exploit these files to store malicious behavior.

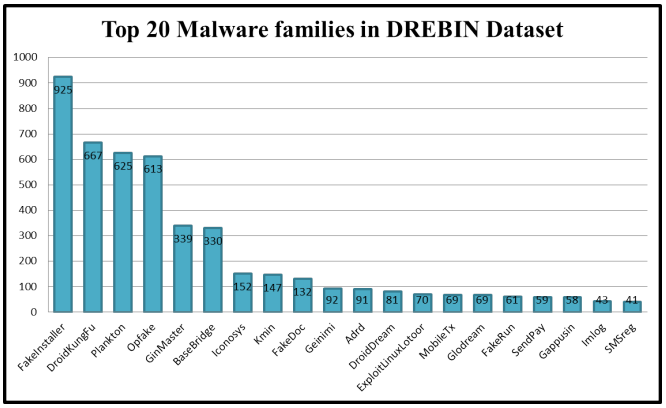

FIGURE 1. Count of samples in each malware family in DREBIN dataset

The functionality and purpose of files and folders within an apk as shown in Figure 3 are explained below:

a) **Meta-Inf /:** It contains the signature files such as CERT.SF and CERT.RSA. It also contains the manifest file i.e. MANIFEST.MF

b) **assets/:** AssetManager object is used by the application to retrieve the application assets detailed in assets folder.

c) **res/:** This folder includes description of resources. These resources are not compiled in resources.arsc folder.

d) **lib/:** The software layer of a processor is associated with a particular type of compiled code that is stored inside this folder.

e) **resources.arsc :** The compiled apk resources are contained within this file. Strings, styles and the paths of images/layout files are part of this content. Data is processed in XML format only.

f) **classes.dex :** Class files are generated after compilation of the java code. These class files are merged into one single dex file using some standard dex tool. Classes.dex contains the Dalvik bytecode. Dalvik Virtual Machine executes the dex file. Any change in dex file will affect the APK.

g) **AndroidManifest.xml:** It includes the set of permissions required by an application, hardware or software components, and linking of API libraries. It also reveals the SDK version.

B. CONVERTING MALWARE APK INTO GRAYSCALE IMAGES

Primarily four types of files such as classes.dex, resource, manifest, and certificate files constitute a stable APK structure [39]. The malware binary bits are paired into 8-bit vectors and in this manner converted over into grayscale images. There are a couple of key advances associated while transforming any malware binary samples into grayscale images. The whole malware substring can be viewed as the grouping of a few substrings. Every substring in a binary code which is 8-length long termed as a pixel. The 8-bit length number stream can be further converted to represent decimal numbers within the range 0 to 255. After the computation of unsigned decimal numbers, the malicious code matrix needs to be generated. All malware executable substrings are further split into 1D vectors of decimal numbers. A one-dimensional vector space can be considered as linear vector space. It is further processed to form a two-dimensional matrix of specific width. Furthermore, some generalizations have been made based on empirical observations. We have fixed the grayscale image widths as indicated by the image size in Table 2 [40]. In this paper, we have used the DREBIN dataset for malware classification purposes. The malware executables of twenty families were converted into grayscale images by following the above-mentioned steps. The illustration of malware images of families such as FakeInstaller, DroidKungFu, Plankton, and Opfake is depicted in Figure 4. These grayscale images relate to various areas of the APK. We have created the images using fifteen unique combinations of Android file structures. Figure 6 summarizes the number of malware images generated for every unique file structure combination. It can been visually interpreted from the figure that the malware images are distributed familywise. CR stands for the certificate file, AM, RS, CL stands for Android manifest file, resources file, and classes.dex file of any malware APK. In Figure 4, the malware images of the families are generated using file combination CR and RS. In Figure 5, the malware images of the families are generated using file combination CR+AM+RS+CL. The variants of the mentioned malware families were found to be dissimilar in their texture. These images found to have different grayscale image textures when generated using different file structure combinations. The texture tends to change with the contents of the malware APK. The malware images generated are different in size. The height of the images is adjusted according to the file size of the malware sample. This motivates to classify and analyze the malware based on malware images.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

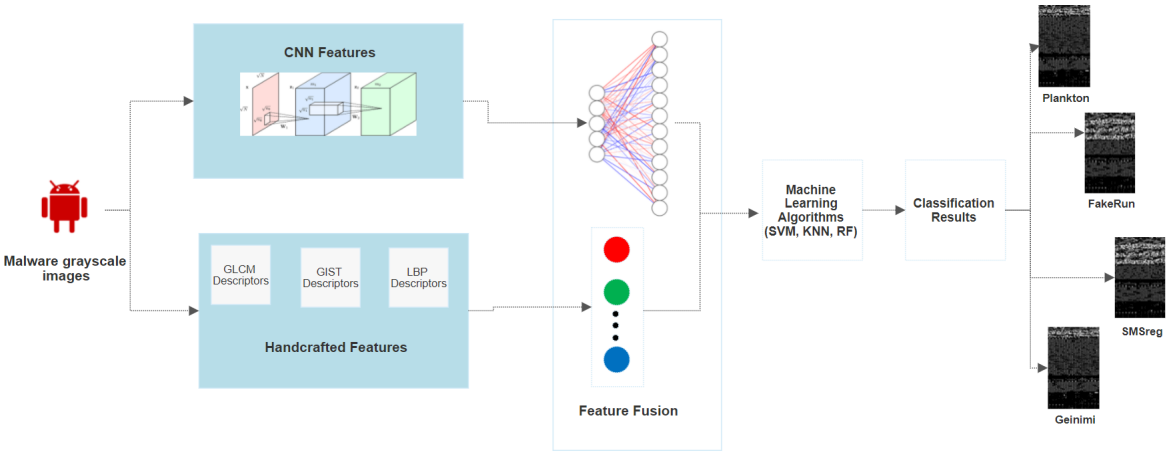

FIGURE 2. The proposed methodology

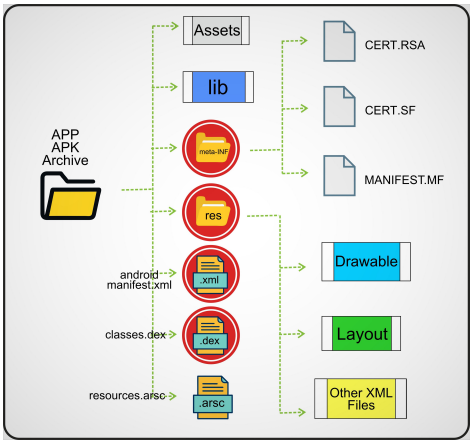

FIGURE 3. Primary resources of an APK

TABLE 2. Fixation of image width

File Size	Width
<50 KB	64
50KB~100KB	128
100KB~200KB	256
200KB~500KB	512
500KB~1000KB	1024

C. EXPERIMENT DESIGN

As depicted in Figure 4 and 5, we can see that malware images have textures in it. The texture is the description of the spatial arrangement of color, intensities, or a selected region in an image. Image texture is a function of spatial variation in pixel intensity which reveals how the pixel values are changing over an area. It eventually defines the visual interpretation of an image. Nowadays, in the era of digital image analysis, textures of the image are used for various purposes such as image segmentation, image classification, texture synthesis, and shapes that can be discriminated using textures. Texture involves the spatial distribution of gray levels. There are multiple uses of textures. Application areas of textures are multidisciplinary such as the food processing industry, biometrics analysis (matching fingerprint, iris, or retina), medical image analysis, remote sensing data analysis (geographic information system), cybersecurity. The texture features are calculated using a statistical approach. The statistical approach includes methods such as GIST, Gray Level Co-occurrence Matrix-based (GLCM) features, and Local Binary Pattern (LBP) features. The stated descriptors are explained as below:

1) Gray Level Co-occurrence Matrix-based (GLCM features)

GLCM are one of the most popular texture features which have been utilized widely for content-based image retrieval, medical image classification, and object recognition. In this approach, texture information from the image is extracted

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

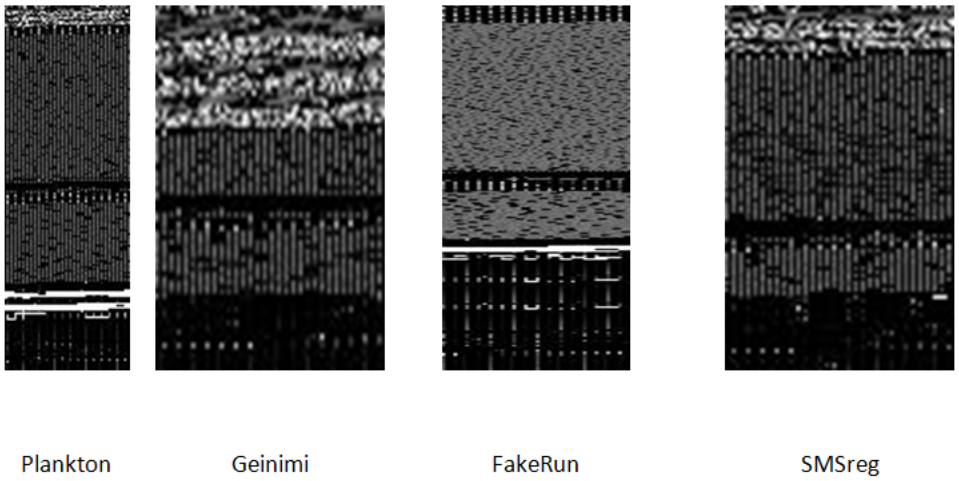

FIGURE 4. Malware images generated using files certificate (CR) and resource (RS) of an Android application

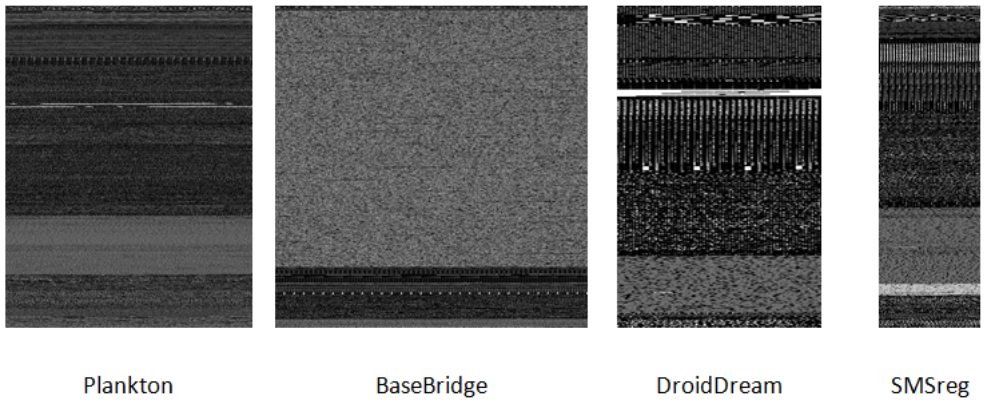

FIGURE 5. Malware images generated using files Android manifest (AM), certificate (CR), classes.dex (CL), and resource (RS) of an Android application

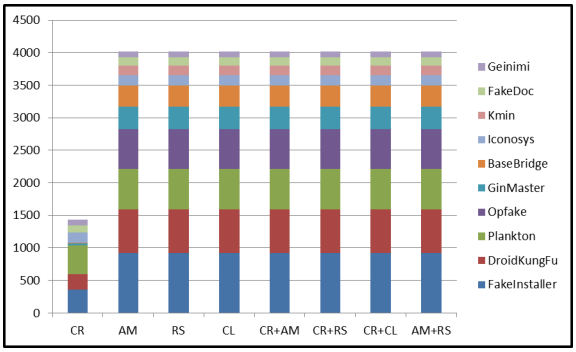

FIGURE 6a. Distribution of eight types of unique malware images across malware families

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

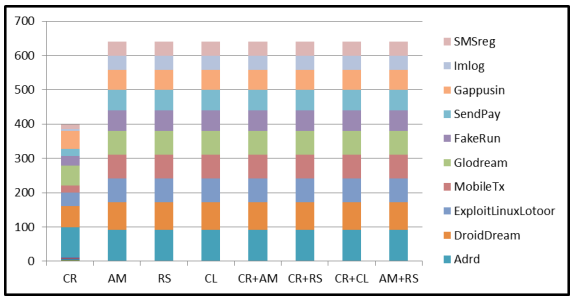

FIGURE 6b. Distribution of eight types of unique malware images across malware families

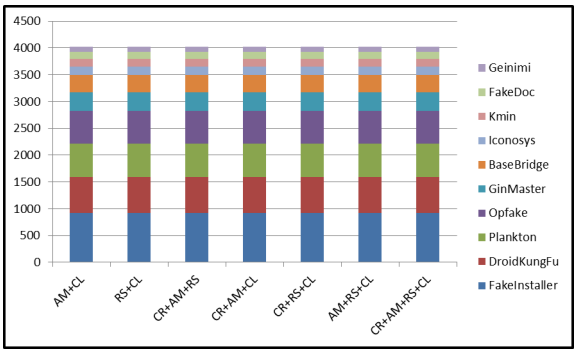

FIGURE 6c. Distribution of seven types of unique malware images across malware families

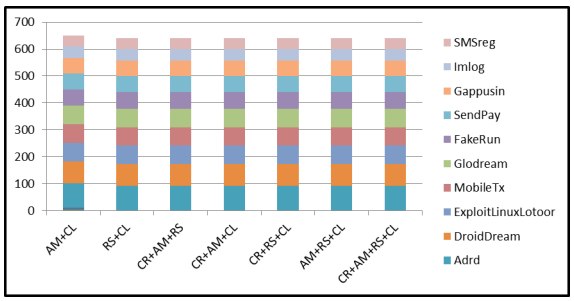

FIGURE 6d. Distribution of seven types of unique malware images across malware families

FIGURE 6. Distribution of malware images across malware families (CR is Certificate, AM is AndroidManifest, RS is Resource, CL is Classes.dex)

from the spatial relationship between the pixels. This spatial relation between the image pixels is defined in terms of distance and orientation. Initially, the GLCM matrix is calculated which estimates the probability density function of the gray level pairs in an image with some specific spatial relationship. The most common choice of distance is 1 in four directions (0°, 45°, 90°, and 135) [22], [37]. Then, several statistics are calculated from this matrix to describe the texture in an image. In this work, only nineteen statistics are used to represent the texture of malware images. These features include contrast, correlation, energy, entropy, homogeneity, sum of square, sum average, sum variance, sum entropy, difference variance, difference entropy and Information measure of correlation. These features are measured for each combination of distance and orientation which results in total 76 features.

2) Local Binary Pattern (LBP)

Local Binary Pattern (LBP) [41], [42] texture descriptor is calculated on malware grayscale images. In a small patch/matrix of the image, the center pixel is surrounded by the neighbors. If the neighbor has the value greater than the center value, it would be replaced with 1 otherwise with 0. For example, consider the 3x3 matrix, there would be 8 neighbors around the center pixel and hence 8-bit sequence would be generated. For every 8-bit sequence, there are 8 such rotations and also there is an integer representation associated with each rotation. LBP is typically defined as the integer value of the minimum of rotations. The parametric value of radius is taken to be 8.

3) Global Image deScripTors (GIST)

GIST algorithm [43], [44] is known for its good accuracy in computer vision tasks. GIST uses 8 orientation of parameters per scale in 4 different blocks. It convolved the image with 32 Gabor filter at 4 scales and 8 orientation to produce 32 feature maps of the same size of an input image. It divides each feature map into 16 regions (4x4 grid) and then averages the feature values within each region. It then concatenates the 16 averaged values of all 32 feature maps resulting in a 16x32=512 GIST descriptors.

Feature fusion has been widely adopted by researchers for detection and classification tasks relevant to computed vision [45]–[49]. We have used GIST descriptors with default value. Using default GIST values produced 512 features in total. For GLCM and LBP, a total of 76 and 58 features respectively are used. CNN architecture produces the vector of length 4096 features. In our work, we have used the concatenation method for feature fusion [50], [51]. Features are concatenated column-wise. The working of CNN architecture has been elaborated in Figure 9.

IV. RESULTS

Top twenty classes with maximum number of instances in the DREBIN dataset were included in this experiment. We have used a handcrafted and CNN feature extraction approach to solve the malware classification problem. The results obtained from the experiments are discussed in the subsequent sections.

A. CLASSIFICATION PERFORMANCE WITH HANDCRAFTED FEATURES

Table 3 shows the classification results with classifiers SVM, KNN, and RF for malware images using three texture descriptors GLCM, GIST, and LBP. To identify the effectiveness of the proposed solution, various evaluation measures such as Accuracy, Precision, Recall, and Error Rate were explored. We have used the default parameters of machine classifiers - Support Vector Machines, K-Nearest Neighbor, and Random Forest which are mentioned in the Scikit-Learn library. The important findings from the outcomes are as follows.

The performance of SVM classifier degrades for malware images when used with GLCM and LBP texture descriptors. As shown in Table 3, when features were extracted using the GLCM algorithm and used SVM to perform classification, the accuracy for 11 different combinations of image sections lies only between 51% to 59%. For some combination of image sections, it is even worse. For combination RS, CR+RS, AM+RS, CR+AM+RS, it is 30.56%, 29.71%, 38.62%, 39.21% respectively. In LBP+SVM classification results, the accuracy for 14 combinations of image sections lies only between 54% to 63%. For RS combination, it is even poorer which is 46.62%. The performance of the SVM classifier significantly improved when used with GIST text descriptors. For 15 unique combinations of image sections, the accuracy lies between 82% to 92%. The highest accuracy of 91.29% was observed for combination CR+AM. The lowest accuracy of 82.92% was observed for combination CR.

When the KNN classifier was used to classify GLCM features, a decent accuracy between 79% to 84% was observed for most of the combinations of image sections. The highest accuracy of 83.36% was observed for the combination of CR+AM. The poor performance was seen against only one combination i.e. CR with accuracy 56.05%. The KNN classifier also performs well when used with LBP descriptors. The accuracy for 14 out of 15 combinations lies between 79% to 86%. For combination RS+CL malware images, the highest accuracy of 85.18% was recorded. It is closely followed by malware images combination of CR+RS+CL with an accuracy of 84.92%. The lowest accuracy of 61.53% was observed for CR malware images. The classification results using GIST-KNN, showed good accuracy which is comparable to classification results of GIST-SVM. For 14 combinations, accuracy lies between 85% to 91%. The highest accuracy of 90.12% was observed for combination CR+AM. The lowest accuracy of 80.60% was observed for CR.

The better performance was seen in the results when GLCM features are extracted from malware images and classified using RF. For most of the combination of image file sections, the accuracy lies between 87% to 92%. The highest accuracy of 91.74% was observed for the combination of CR+AM of malware images. The lowest accuracy of 76.62% was observed for CR malware images. When malware images are classified using GIST texture descriptors + RF classifier, the classification results are decent but not better than GIST-SVM and GIST-KNN. The accuracy for all combinations of malware images lies between 83% to 89%. The highest accuracy of 88.69% was observed for the combination of CR+AM of malware images. The lowest accuracy of 83.42% was observed for CR.

When features were extracted using LBP texture descriptors and classified using RF, it provided better classification results than LBP-SVM and LBP-KNN. Most of the combinations of malware image sections attain the accuracy between 84% to 86%.

The top average accuracy observed to be 88.05%, 87.99%, 87.44%, 85.32%, and 84.37% for GIST-SVM, GLCM-RF, GIST-KNN, GIST-RF, LBP-RF respectively. The classification results with all classifiers SVM, KNN, and RF on all combinations of malware image sections using GIST algorithm found to be maximum stable. GIST features are more helpful in drawing the original semantics and analysis of the malware image. GLCM features when classified with SVM classifier shown poorer performance with an average accuracy of 48.35%. LBP features performed well with classifiers KNN and RF with an average accuracy of 81.17% and 84.37% respectively. LBP texture descriptors did not perform well with the SVM classifier and attain an average accuracy of 55.87%. CR+AM malware images have attained the maximum accuracy.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

TABLE 3. Accuracy of handcrafted features on fifteen combination of malware images

Image Combination	GLCM-SVM	GLCM-KNN	GLCM-RF	GIST-SVM	GIST-KNN	GIST-RF	LBP-SVM	LBP-KNN	LBP-RF
CR	51.08%	56.05%	76.62%	82.92%	80.60%	83.42%	56.22%	61.53%	67.66%
AM	57.02%	81.60%	89.86%	90.70%	89.01%	86.35%	59.23%	79.39%	85.37%
RS	30.56%	79.45%	87.84%	86.02%	85.44%	84.07%	46.62%	80.88%	86.80%
CL	52.86%	82.25%	88.69%	88.69%	88.88%	86.28%	58.32%	83.62%	85.11%
CR+AM	58.45%	83.36%	91.74%	91.29%	90.12%	88.69%	62.42%	81.99%	86.22%
CR+RS	29.71%	79.91%	87.52%	85.70%	85.63%	83.81%	46.75%	79.97%	84.46%
CR+CL	53.06%	81.66%	89.08%	88.69%	89.14%	85.63%	58.19%	84.46%	85.11%
AM+RS	38.62%	80.17%	88.43%	87.26%	86.28%	84.14%	52.99%	79.52%	85.83%
AM+CL	52.86%	82.70%	88.56%	88.95%	88.49%	85.83%	58.45%	84.46%	86.35%
RS+CL	52.21%	82.31%	88.30%	88.75%	88.56%	85.18%	56.76%	85.18%	85.89%
CR+AM+RS	39.21%	80.95%	88.43%	87.39%	85.89%	84.79%	54.23%	79.97%	84.46%
CR+AM+CL	52.73%	83.09%	88.82%	87.84%	88.82%	85.31%	58.52%	84.07%	85.31%
CR+RS+CL	52.28%	81.86%	88.62%	89.01%	88.23%	85.24%	57.35%	84.92%	86.09%
AM+RS+CL	52.21%	82.18%	88.82%	88.56%	88.04%	85.37%	55.98%	83.94%	85.57%
CR+AM+RS+CL	52.34%	82.05%	88.56%	88.95%	88.49%	85.70%	56.05%	83.62%	85.37%

B. IMPACT OF NORMALIZATION ON CLASSIFICATION PERFORMANCE

In this work, the Min-Max Normalization method is considered to investigate the impact of data normalization on the classification performance of malware images. The method scales the un-normalized data to a predefined lower and upper bounds linearly. The data is usually rescaled within the range of 0 to 1 or -1 to 1. Table 4 shows the classification accuracy for normalized using handcrafted features. The classification performance on normalized data is discussed below.

The classification results produced using GLCM features and classifiers SVM, KNN, and RF are depicted in Figure 7(a), 7(b), and 7(c). The difference in the classification results with normalized and unnormalized data can be seen visually. Figure 7(a) shows that there is an improvement in classification performance with normalized data. For all combinations of the malware image section, the accuracy is observed to be significantly improved using GLCM-SVM. It is to be worth noted that GLCM-SVM showed the worst performance on unnormalized data with an average classification accuracy of 48.35%. But with normalized data, the average classification accuracy of GLCM-SVM improved to 84.33%. The highest accuracy of 92.20% was observed with GLCM-SVM using AM malware images. Therefore, min-max normalization proved to be substantial to make GLCM-SVM a more stable model.

Normalization not always improve classification performance. Incase of GLCM-RF, the classification accuracy is observed to be declined for some combination of malware image sections. But it also increased for some of the combinations. The maximum fall in accuracy is seen to be 0.85% for combination CR+RS. The maximum increase in accuracy is seen to be of 0.85% for RS+CL. Hence, we can say that normalization does not have a significant impact on the combination GLCM-RF as shown in Figure 7(b)

The classification results improved with normalized data using GLCM-KNN. It got improved for all combinations of image file sections. There is an increase in accuracy ranges from the window of 6.57% to 8.39% for at least fourteen combinations as shown in Figure 7(c). It is observed that classification accuracy for CR malware images has been increased by 22.55%. Earlier GLCM-KNN with unnormalized data was the worst performer on CR with an accuracy of 56.05% but with normalized data, it got increased to 78.61%. The average classification accuracy of GLCM-KNN got improved from 79.97% to 87.91% due to the impact of normalization.

There is no major impact of normalization was observed on classification accuracy obtain using GIST features and classifiers SVM, KNN, and RF.

The significant improvement is observed in the classification results when malware images were classified using LBP-SVM as shown in Figure 8(a). At least an increase of 21% to 27% in the accuracy has been observed in most of the combinations of malware image sections. The average classification accuracy increases from 55.87% to 80.37%. Thus, normalization makes the LBP-SVM a more stable model.

Normalization also improved the classification results of LBP-KNN as shown in Figure 8(b). The average classification results of LBP-KNN with normalized data observed to be 84.34%.

There is no significant improvement observed in the results of LBP+RF with normalized data as shown in Figure 8(c).

C. FEATURE FUSION OF CNN AND HANDCRAFTED FEATURES ON NORMALIZED DATA

For feature fusion experiments, we have combined CNN features and handcrafted features to perform Android malware image classification. The CNN architecture used in this work was adopted from [7]. The classification is performed using SVM, KNN, and RF classifiers with normalized data. The graphical representation of the CNN architecture is presented in Figure 9. The grayscale images are fed into the CNN architecture. CNN will extract the features from the malware images. Conv2D and MaxPooling2D are the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

TABLE 4. Accuracy of handcrafted features on fifteen combination of malware images (normalized dataset)

Image Combination	GLCM-SVM	GLCM-KNN	GLCM-RF	GIST-SVM	GIST-KNN	GIST-RF	LBP-SVM	LBP-KNN	LBP-RF
CR	78.94%	78.61%	77.11%	83.58%	80.60%	83.25%	63.02%	65.01%	68.99%
AM	92.20%	89.99%	90.12%	90.77%	89.66%	86.35%	80.30%	83.49%	85.11%
RS	79.13%	86.54%	87.58%	84.98%	85.37%	84.85%	73.93%	83.88%	86.35%
CL	84.85%	88.36%	88.62%	88.82%	88.62%	85.63%	84.01%	85.70%	85.24%
CR+AM	90.96%	90.57%	91.81%	91.35%	89.21%	88.56%	83.88%	84.72%	86.09%
CR+RS	81.27%	86.15%	86.67%	86.09%	85.37%	84.79%	76.92%	84.07%	84.14%
CR+CL	83.49%	89.34%	88.75%	88.75%	88.56%	85.83%	84.20%	87.06%	86.15%
AM+RS	83.42%	87.13%	88.36%	87.26%	86.35%	84.07%	77.96%	86.09%	86.22%
AM+CL	83.75%	89.40%	88.69%	88.82%	88.43%	85.89%	84.66%	87.26%	85.76%
RS+CL	84.07%	88.95%	89.14%	89.01%	88.43%	85.89%	83.75%	86.54%	85.89%
CR+AM+RS	86.28%	88.17%	88.49%	87.58%	86.35%	84.40%	76.72%	85.05%	84.59%
CR+AM+CL	84.66%	89.21%	88.43%	88.04%	88.75%	84.66%	84.92%	87.13%	86.02%
CR+RS+CL	83.68%	88.62%	88.56%	89.53%	88.10%	85.18%	83.22%	86.67%	86.09%
AM+RS+CL	84.14%	89.01%	89.08%	88.82%	88.43%	85.11%	84.20%	86.15%	85.70%
CR+AM+RS+CL	84.07%	88.62%	88.95%	89.21%	88.62%	85.37%	83.94%	86.35%	85.31%

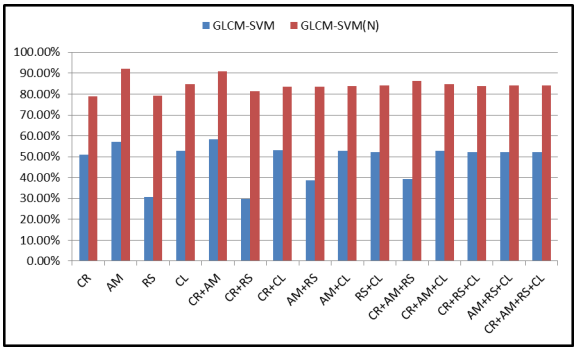

FIGURE 7a. Comparison of Accuracy of GLCM-RF on fifteen combination of malware images of normalized and un-normalized dataset

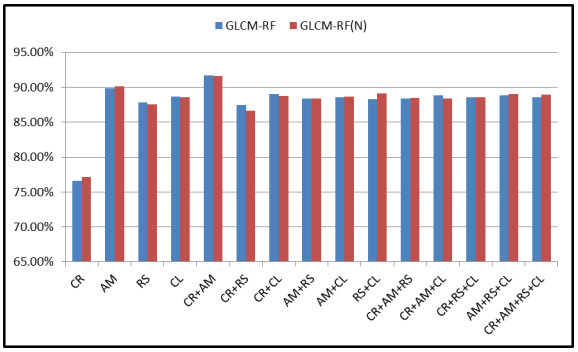

FIGURE 7b. Comparison of Accuracy of GLCM-RF on fifteen combination of malware images of normalized and un-normalized dataset

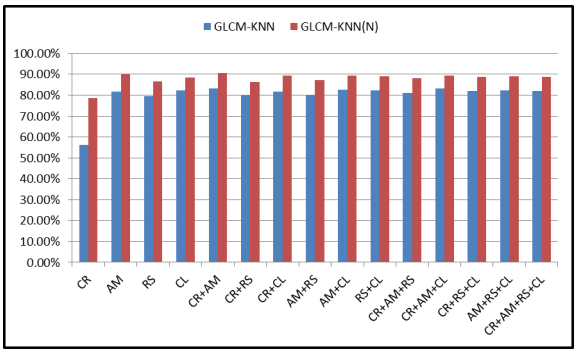

FIGURE 7c. Comparison of Accuracy of GLCM-KNN on fifteen combination of malware images of normalized and un-normalized dataset

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

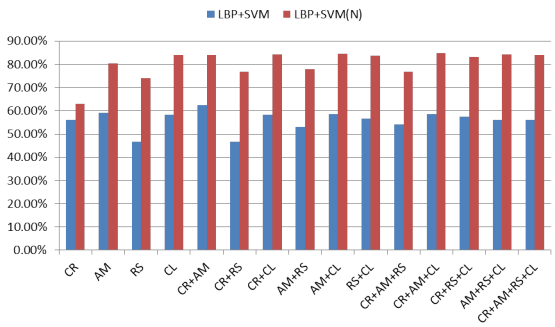

FIGURE 8a. Comparison of Accuracy of LBP-SVM on fifteen combinations of grayscale malware images ofnormalized and un-normalized dataset

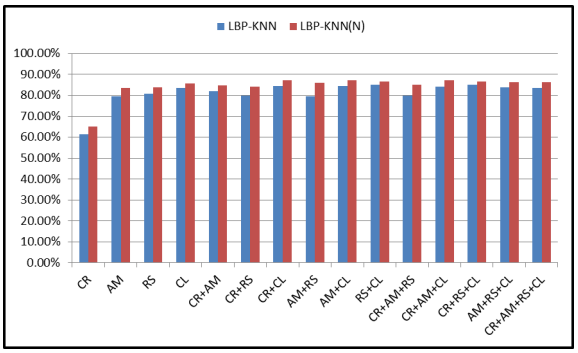

FIGURE 8b. Comparison of Accuracy of LBP-KNN on fifteen combinations of grayscale malware images ofnormalized and un-normalized dataset

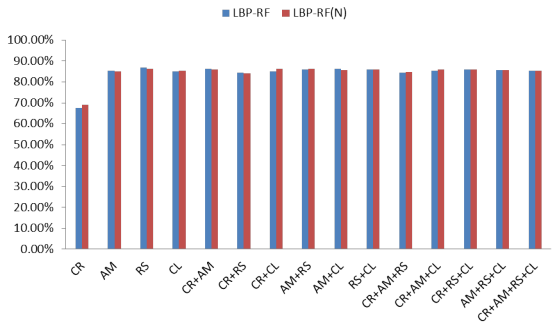

FIGURE 8c. Comparison of Accuracy of LBP-RF on fifteen combinations of grayscale malware images ofnormalized and un-normalized dataset

other two libraries imported to set up the environment for neural networks. MaxPooling will help to reduce the size of the image. Other libraries imported are Activation, Dropout, Flatten, and Dense. Malware grayscale images are in two dimensions. The height, width, and depth of the input image are taken to be 108, 108, and 1 respectively. To build and train the CNN on the malware images of different families, we added the three convolutional layers to the model which are represented as the Conv2D (32,7,7), Conv2D (128,5,5), and Conv2D (256,3,3). The first argument defines the number of output filters in the convolution layer. The next two arguments define the kernel size. Kernel size is a tuple of two integers that is specifying the width and height of the two-dimensional convolutional window. ReLu is used as the activation layer in the CNN architecture. Max-pooling layer has been deployed with pool size 3X2, 3X3, and 2X2 after each convolution layer. To avoid the overfitting problem, a dropout layer with a value of 0.5 was used. Three dense layers with 50, 100, 200 neurons were deployed in the network. The softmax activation function is added to the output layer of 20 neurons.

1) Comparison with GLCM-SVM, GIST-SVM, LBP-SVM

Feature Fusion with SVM classifier significantly improves the classification accuracy when compared with the results of GLCM-SVM. A decent hike of 7% to 10% was observed for at least ten combinations of malware image sections as depicted in Figure 10(a). These combinations are AM+RS+CL, CR+AM+CL, CR+AM+RS+CL, AM+CL, CL, AM+RS, RS+CL, CR+RS+CL, CR+CL, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

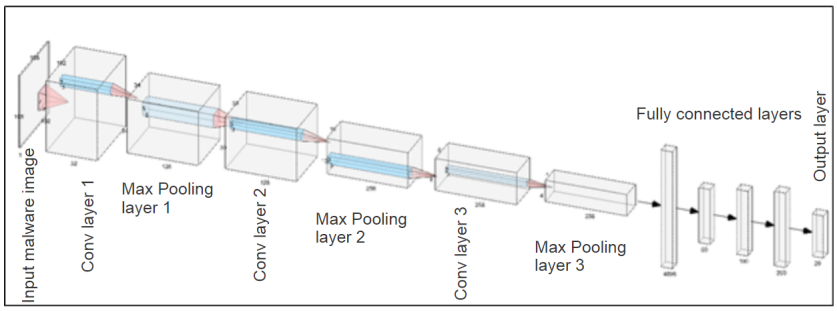

FIGURE 9. The CNN architecture

CR+RS with increased accuracy 6.70%, 6.96%, 7.02%, 7.35%, 7.35%, 7.35%, 7.61%, 8.13%, 8.45%, and 9.23% respectively. For the rest of the combinations of malware image sections, the classification accuracy increased from at least 2% to the maximum of 7%. A slight decline of 0.65% was observed in classification accuracy for malware images created using AM file. The highest accuracy of 93.24% was observed for combination CR+AM using Feature Fusion-SVM classifier.

The classification results of Feature Fusion-SVM are also compared with GIST-SVM as shown in Figure 10(b). It has been observed that classification results produced using Feature Fusion-SVM are better than the results of GIST-SVM on various combinations of malware image sections. For at least thirteen combinations, there is a hike in classification accuracy between the range 1% to 4%.

Figure 10(c) revealed that Feature Fusion-SVM also outperformed the combination LBP-SVM. For combinations AM, AM+RS, CR+RS, RS, CR+AM+RS the accuracy increased by 11.25%, 12.81%, 13.59%, 14.69%, and 15.47% respectively. For the rest of the combinations, there was an increase in accuracy between the range 6% to 10%. The average accuracy is observed to be 90.90% using Feature Fusion-SVM whereas it was 80.37% using LBP-SVM.

2) Comparison with GLCM-KNN, GIST-KNN, LBP-KNN

The comparison results of Feature Fusion-KNN with GLCM-KNN, GIST-KNN, LBP-KNN are shown in Figure 11(a), 11(b), and 11(c). An increase of 0.39% to 2.86% in classification accuracy was observed when the results of Feature Fusion-KNN are compared with GLCM-KNN. On the other hand, an increase of 0.83% to 3.77% in classification accuracy was observed when the results of Feature Fusion-KNN are compared with GIST-KNN. KNN outperformed the results of LBP-KNN when it was used with feature fusion. It was observed that LBP-KNN showed the worst performance against CR malware images with an accuracy of 65.01%. KNN performance on CR malware images got better with Feature Fusion and obtain an accuracy of 81.43%. For the rest of the combinations, accuracy ranges from 2% to 7%.

3) Comparison with GLCM-RF, GIST-RF, LBP-RF

The comparison results of Feature Fusion-RF with GLCM-RF, GIST-RF, LBP-RF are depicted in Figure 12(a), 12(b), 12(c). There is no significant difference between the results which are produced by GLCM-RF and Feature Fusion-RF. The variation in accuracy is observed when results of GIST-RF are compared with classification results of Feature Fusion-RF. An increase in accuracy between the range of 0.17% to 4.16% was observed for RF when used in linear combination with Feature Fusion. The results of Feature Fusion-RF are also better than the results of LBP-RF. The average classification result of LBP-RF is recorded as 84.51% whereas it is 88.34% when RF is used with handcrafted and CNN features.

The top five type of malware images against which handcrafted features and feature fusion strategy have attained maximum accuracy are depicted in Figure 13(a), 13(b) ,13(c), 13(d), 13(e). It revealed that classifiers have attained the maximum accuracy on AM and CR+AM malware images. It is observed that sharp spikes appear when using the feature fusion strategy for the classification of Android malware images. For CR+AM malware images, GLCM, GIST, and LBP features attained an average accuracy of 91.05%, 89.71%, and 84.89% respectively whereas feature fusion strategy attained an accuracy of 91.3%. For AM malware images, GLCM, GIST, and LBP features attained an average accuracy of 90.77%, 88.93%, and 82.96% respectively whereas feature fusion strategy attained an accuracy of 89.62%. For CR+CL malware images, GLCM, GIST, and LBP features attained an average accuracy of 87.19%, 87.71%, and 85.80% respectively whereas feature fusion strategy attained an accuracy of 90.75%. For AM+CL malware images, GLCM, GIST, and LBP features attained an average accuracy of 87.28%, 87.71%, and 85.89% respectively whereas feature fusion strategy attained an accuracy of 90.42%. For CR+AM+CL malware images, GLCM, GIST, and LBP features attained an average accuracy of 87.43%, 87.15%, and 86.02% respectively whereas feature fusion strategy attained an accuracy of 90.31%.

The confusion matrix for the twenty malware families is shown in Figure 14. The performance metrics such as precision, recall, and error rate is also shown in Figure 15. As discussed earlier, Feature Fusion with SVM classifier

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

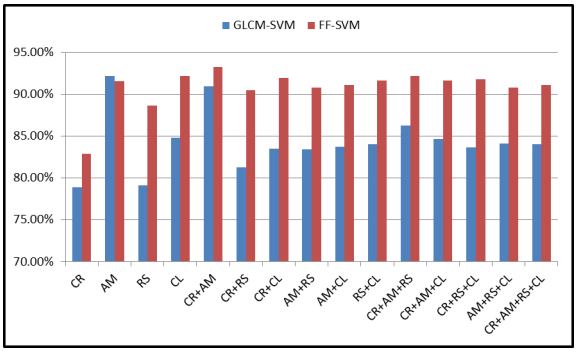

FIGURE 10a. Comparison of Accuracy of GLCM-SVM and Feature Fusion-SVM on fifteen combinations of grayscale

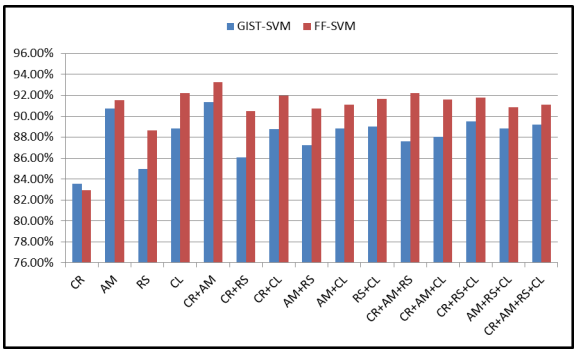

FIGURE 10b. Comparison of Accuracy of GIST-SVM and Feature Fusion-SVM on fifteen combinations of grayscale

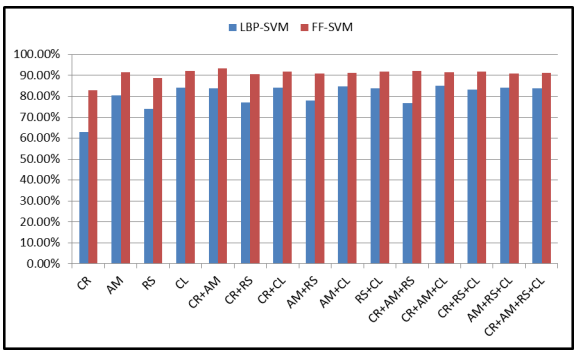

FIGURE 10c. Comparison of Accuracy of LBP-SVM and Feature Fusion-SVM on fifteen combinations of grayscale

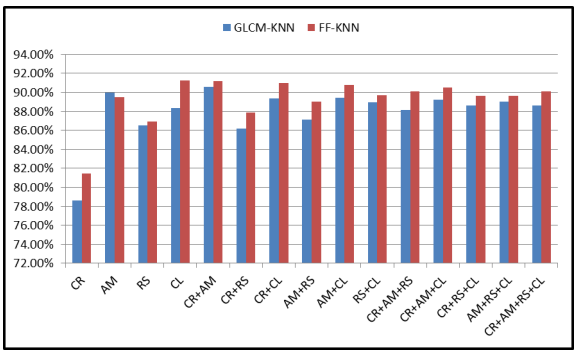

FIGURE 11a. Comparison of Accuracy of GLCM-KNN and Feature Fusion-KNN on fifteen combinations of grayscale

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

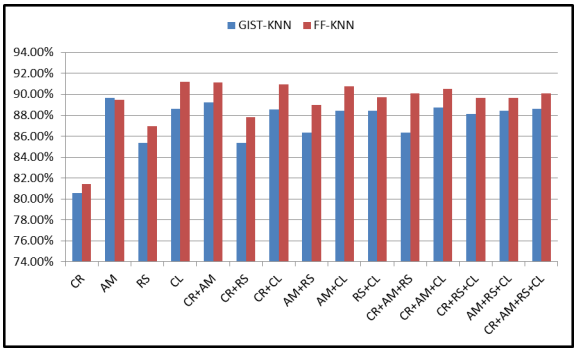

FIGURE 11b. Comparison of Accuracy of GIST-KNN and Feature Fusion-KNN on fifteen combinations of grayscale

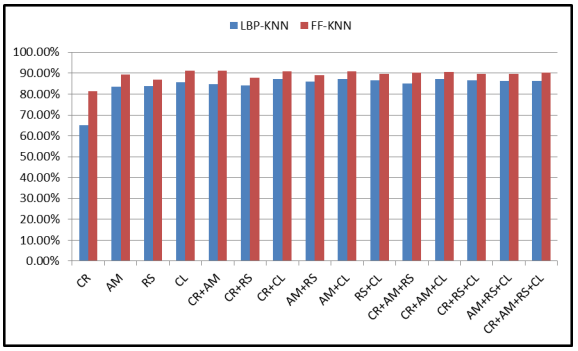

FIGURE 11c. Comparison of Accuracy of LBP-KNN and Feature Fusion-KNN on fifteen combinations of grayscale

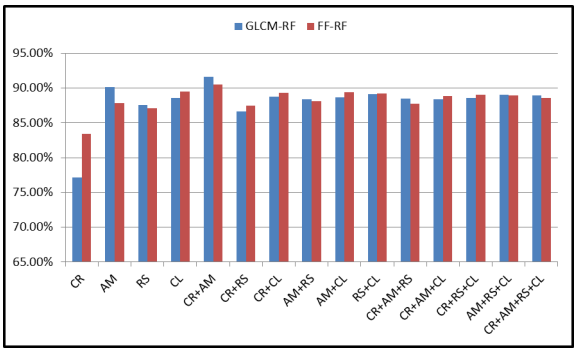

FIGURE 12a. Comparison of Accuracy of GLCM-RF and Feature Fusion-RF on fifteen combinations of grayscale

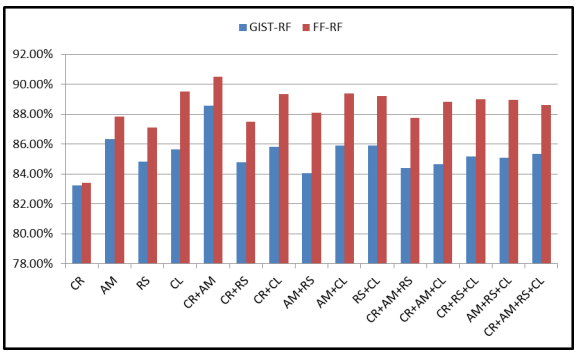

FIGURE 12b. Comparison of Accuracy of GIST-RF and Feature Fusion-RF on fifteen combinations of grayscale

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

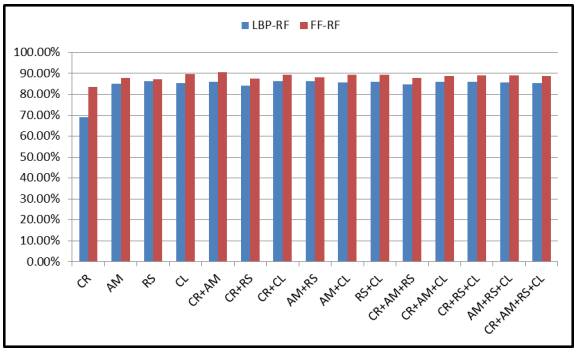

FIGURE 12c. Comparison of Accuracy of LBP-RF and Feature Fusion-RF on fifteen combinations of grayscale

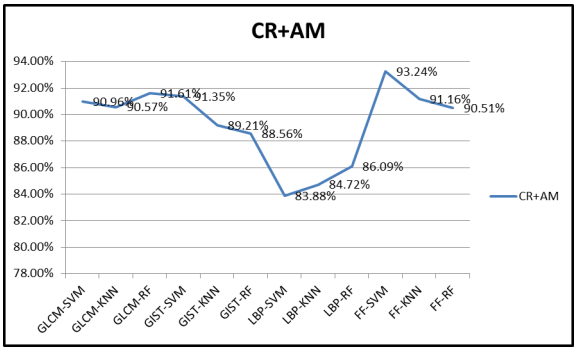

FIGURE 13a. Graphplot showing accuracy of handcrafted features and feature fusion against CR+AM malware image

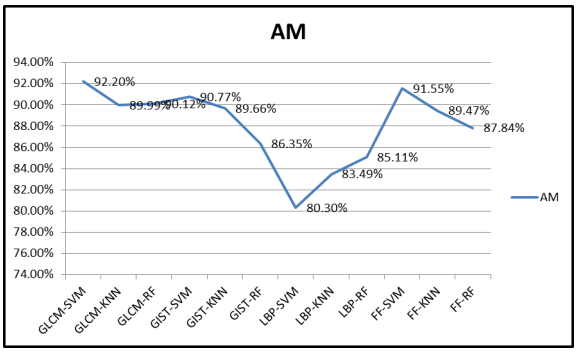

FIGURE 13b. Graphplot showing accuracy of handcrafted features and feature fusion against AM malware image

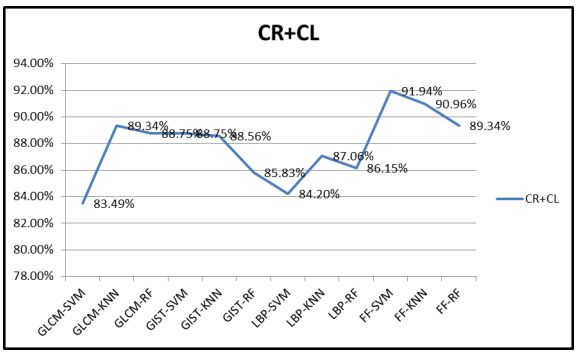

FIGURE 13c. Graphplot showing accuracy of handcrafted features and feature fusion against CR+CL malware image

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

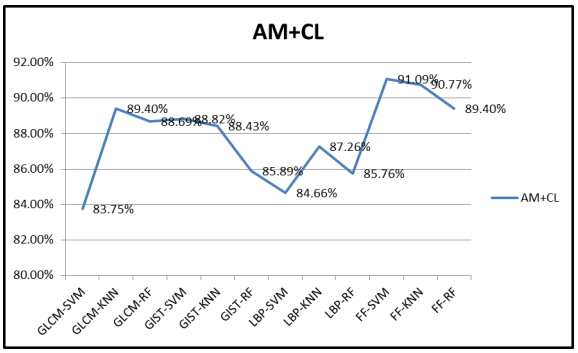

FIGURE 13d. Graphplot showing accuracy of handcrafted features and feature fusion against AM+CL malware image

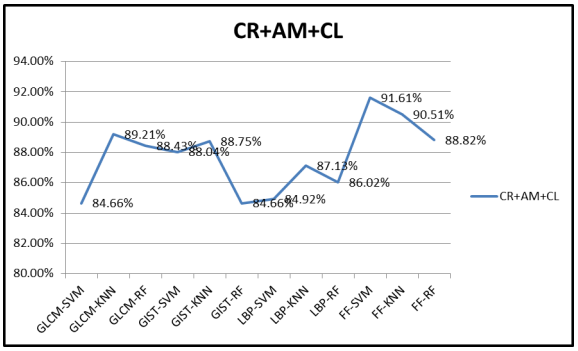

FIGURE 13e. Graphplot showing accuracy of handcrafted features and feature fusion against CR+AM+CL malware image

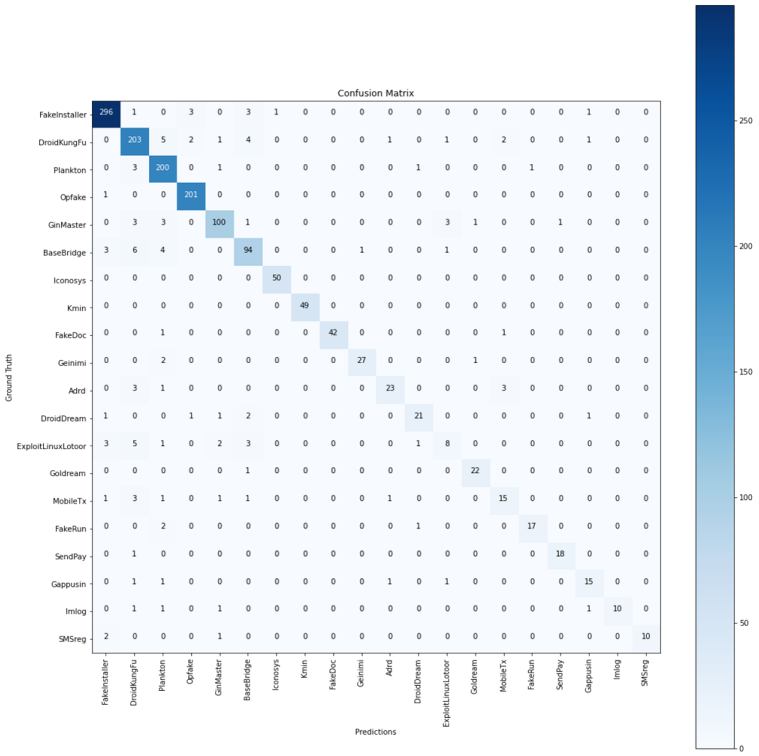

FIGURE 14. A confusion matrix

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3090998, IEEE Access

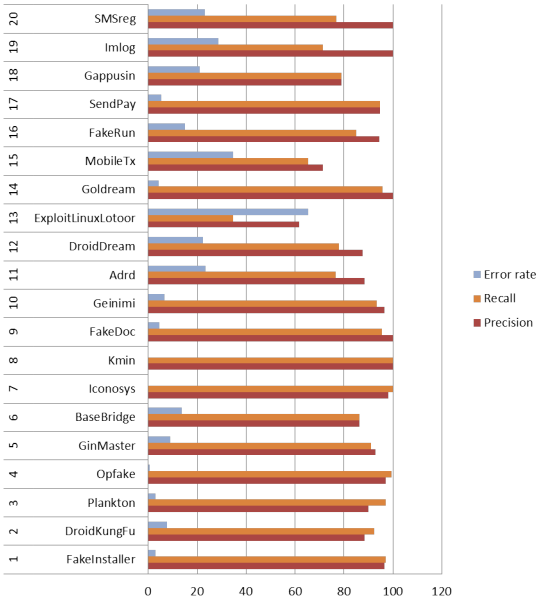

FIGURE 15. Performance metrics obtained using Feature Fusion-SVM classifier

achieved the highest accuracy of 93.24% using CR+AM malware images. Among the total malware families, the families such as Kmin, GoldDream, FakeDoc, Iconosys, Opfake, and FakeInstaller attain high precision and recall. ExploitLinuxLootor, MobileTx, Gappusin, and BaseBridge are the families against which low precision and recall were observed. The performance degrades due to the less number of samples in these families.

The error rate for the malware family ExploitLinuxLotoor is found to be 65.21%, which is relatively high. For MobileTx and Imlog, it was 34.78% and 28.57% respectively. The error rate for the families Adrd, SMSreg, DroidDream, and Gappusin varies from 21% to 24%. All the samples for the malware family Kmin were correctly classified. Even no sample of other class gets misclassified to Kmin malware family class. Therefore, its error rate found to be zero. The family Iconosys also attained the error rate of zero but achieve the precision of 98.03%. Only one sample of FakeInstaller gets misclassified into the Iconosys malware family. One sample of Plankton class gets misclassified into SendPay class and one sample of Sendpay gets misclassified into DroidKungFu class. For this reason, malware family Iconosys attain an equal precision and recall rate. The error rate of 2.91% and 2.95% was recorded for malware family Plankton and FakeInstaller respectively. For Opfake family, it was observed to be 0.49%.

V. CONCLUSION AND FUTURE SCOPE

A series of experiments were conducted for the analysis and classification of Android malware images. The handcrafted features used in this work are Gray Level Co-occurrence Matrix (GLCM), Global Image deScripTors (GIST), and Local Binary Pattern (LBP). LBP features do not contain much information for malware classification. GIST features with classifiers SVM, KNN, and RF showed good classification accuracy. Min-max normalization on the dataset showed a great impact on the proposed methodology. GLCM-SVM achieved the highest classification accuracy of 92.20% on AM malware images closely followed by the GLCM-RF and GIST-SVM model that achieved an accuracy of 91.81% and 91.35% respectively on CR+AM malware images. Furthermore, CNN and handcrafted features were fused to form the feature fusion strategy for the classification of Android malware images. The classification results obtained using handcrafted features are compared with results achieved using feature fusion methodology. It was found that the classification performance of all the classifiers eventually increased when feature fusion was deployed. Of the top malware images revealed in this work, feature fusion undoubtedly outperforms handcrafted features in the classification of Android malware images. The highest accuracy of 93.24% was observed for malware image combination CR+AM using the Feature Fusion-SVM classifier. Therefore, the efforts can be saved in inspecting the entire APK structure for the classification of Android malware. The proposed visualization technique based on feature fusion will let the same work done with lesser resources and time. The primary focus of this study, was on the feature fusion technique to identify the descriptors, which could help to differentiate between different types of Android malware families. The study on the correlation of features can be another interesting area to explore. The features that are extracted may contain irrelevant or redundant features. Therefore, as future scope of this work, we tend to deploy suitable feature extraction techniques to identify and remove the redundant features by analyzing the correlation between

them. Moreover, the use of ensemble learning in CNNs and other transfer learning models considering hyperparameters optimization sets the future scope of this work.

ACKNOWLEDGMENT

Jaiteg Singh and Farman Ali contributed equally and are co-first authors.

REFERENCES

[1] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "RiskRanker : Scalable and Accurate Zero-day Android Malware Detection Categories and Subject Descriptors," Proc. 10th Int. Conf. Mob. Syst. Appl. Serv. MobiSys '12, New York, NY, USA, 2012. ACM, pp. 281–294, 2012.

[2] M. F. A. Razak, N. B. Anuar, R. Salleh, and A. Firdaus, "The rise of 'malware': Bibliometric analysis of malware study," J. Netw. Comput. Appl., vol. 75, pp. 58–76, 2016. [Online]. Available: 10.1016/j.jnca.2016.08.022

[3] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," Comput. Secur., vol. 77, pp. 871–885, 2018.

[4] N. Xie, X. Wang, W. Wang, and J. Liu, "Fingerprinting Android malware families," Front. Comput. Sci., vol. 13, no. 3, pp. 637–646, 2019.

[5] A. Sadeghi, H. Bagheri, J. Garcia, and S. Malek, "A Taxonomy and Qualitative Comparison of Program Analysis Techniques for Security Assessment of Android Software," IEEE Trans. Softw. Eng., vol. 43, no. 6, pp. 492–530, 2017. [Online]. Available: 10.1109/TSE.2016.2615307

[6] F. Rustam, M. A. Siddique, H. U. R. Siddiqui, S. Ullah, A. Mehmood, I. Ashraf, and G. S. Choi, "Wireless capsule endoscopy bleeding images classification using cnn based model," IEEE Access, vol. 9, pp. 33 675–33 688, 2021.

[7] 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, 2020.

[8] P. Zegzhda, D. Zegzhda, E. Pavlenko, and G. Ignatev, "Applying deep learning techniques for Android malware detection," pp. 1–8, 2018, in Proceedings of the 11th International Conference on Security of Information and Networks.

[9] M. Ganesh, P. Pednekar, P. Prabhuswamy, D. S. Nair, Y. Park, and H. Jeon, "Cnn-based android malware detection," pp. 60–65, 2017, in 2017 International Conference on Software Security and Assurance (ICSSA).

[10] T. H.-D. Huang and H. Y. Kao, "R2-D2: color-inspired convolutional neural network (CNN)-based android malware detections," in 2018 IEEE International Conference on Big Data (Big Data), pp. 2633–2642, 2018.

[11] X. Xiao, "An image-inspired and CNN-based Android malware detection approach," pp. 1259–1261, 2019, in 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE).

[12] W. Wang, M. Zhao, and J. Wang, "Effective android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," J. Ambient Intell. Humaniz. Comput., vol. 10, no. 8, pp. 3035–3043, 2019.

[13] A. de Oliveira and R. J. Sassi, "Chimera: An Android Malware Detection Method Based on Multimodal Deep Learning and Hybrid Analysis," 2020.

[14] D. Thakur, "Classification of android malware using its image sections," International Journal of Advanced Trends in Computer Science and Engineering, vol. 9, no. 4, pp. 6151–6155, aug 2020.

[15] Y. Ding, X. Zhang, J. Hu, and W. Xu, "Android malware detection method based on bytecode image," J. Ambient Intell. Humaniz. Comput., pp. 1–10, 2020.

[16] J. Singh, T. Gera, F. Ali, D. Thakur, K. Singh, and K.-s. Kwak, "Understanding research trends in android malware research using information modelling techniques," CMC-COMPUTERS MATERIALS & CONTINUA, vol. 66, no. 3, pp. 2655–2670, 2021.

[17] T. Gera, J. Singh, D. Thakur, and P. Faruki, "A semi-automated approach for identification of trends in android ransomware literature," in Machine Learning for Networking: Third International Conference, MLN 2020, Paris, France, November 24–26, 2020, Revised Selected Papers 3. Springer International Publishing, 2021, pp. 265–283.

[18] D. Thakur, T. Gera, and J. Singh, "Android anti-malware techniques and its vulnerabilities: A survey," in Smart Innovations in Communication and Computational Sciences. Springer, 2019, pp. 315–328.

[19] R. U. Khan, X. Zhang, and R. Kumar, "Analysis of ResNet and GoogleNet models for malware detection," J. Comput. Virol. Hacking Tech., vol. 15, no. 1, pp. 29–37, 2019.

[20] L. Shiqi, T. Shengwei, Y. Long, Y. Jiong, and S. Hua, "Android malicious code Classification using Deep Belief Network." KSII Trans. Internet Inf. Syst., vol. 12, no. 1, 2018.

[21] 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, 2011, pp. 1–7.

[22] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-g. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," IEEE Transactions on Industrial Informatics, vol. 14, no. 7, pp. 3187–3196, 2018.

[23] J.-S. Luo and D. C.-T. Lo, "Binary malware image classification using machine learning with local binary pattern," in 2017 IEEE International Conference on Big Data (Big Data). IEEE, 2017, pp. 4664–4667.

[24] A. Makandar and A. Patrot, "Malware analysis and classification using artificial neural network," in 2015 International conference on trends in automation, communications and computing technology (I-TACT-15). IEEE, 2015, pp. 1–6.

[25] S. K. Dash, G. Suarez-Tangil, S. Khan, K. Tam, M. Ahmadi, J. Kinder, and L. Cavallaro, "Droidscribe: Classifying android malware based on runtime behavior," in 2016 IEEE Security and Privacy Workshops (SPW). IEEE, 2016, pp. 252–261.

[26] F. Martinelli, F. Marulli, and F. Mercaldo, "Evaluating convolutional neural network for effective mobile malware detection," Procedia computer science, vol. 112, pp. 2372–2381, 2017.

[27] A. Makandar and A. Patrot, "Wavelet statistical feature based malware class recognition and classification using supervised learning classifier," Oriental journal of computer science and technology, vol. 10, no. 2, pp. 400–406, 2017.

[28] S. Seok and H. Kim, "Visualized malware classification based-on convolutional neural network," Journal of The Korea Institute of Information Security & Cryptology, vol. 26, no. 1, pp. 197–208, 2016.

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

[30] B. N. Narayanan, O. Djaneye-Boundjou, and T. M. Kebede, "Performance analysis of machine learning and pattern recognition algorithms for malware classification," in 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS). IEEE, 2016, pp. 338–342.

[31] S. Choi, S. Jang, Y. Kim, and J. Kim, "Malware detection using malware image and deep learning," in 2017 International Conference on Information and Communication Technology Convergence (ICTC). IEEE, 2017, pp. 1193–1195.

[32] K. Kancherla and S. Mukkamala, "Image visualization based malware detection," in 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS). IEEE, 2013, pp. 40–44.

[33] X. Zhou, J. Pang, and G. Liang, "Image classification for malware detection using extremely randomized trees," in 2017 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID). IEEE, 2017, pp. 54–59.

[34] 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 Networks, vol. 105, p. 102154, 2020.

[35] M. Kalash, M. Rochan, N. Mohammed, N. D. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in 2018 9th IFIP international conference on new technologies, mobility and security (NTMS). IEEE, 2018, pp. 1–5.

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

[37] E. M. Karanja, S. Masupe, and M. G. Jeffrey, "Analysis of internet of things malware using image texture features and machine learning techniques," Internet of Things, vol. 9, p. 100153, 2020.

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

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

[40] 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," Computer Networks, vol. 171, p. 107138, 2020.

[41] C. Zhu, C.-E. Bichot, and L. Chen, "Multi-scale color local binary patterns for visual object classes recognition," in 2010 20th International Conference on Pattern Recognition. IEEE, 2010, pp. 3065–3068.

[42] ——, "Image region description using orthogonal combination of local binary patterns enhanced with color information," Pattern Recognition, vol. 46, no. 7, pp. 1949–1963, 2013.

[43] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," International journal of computer vision, vol. 42, no. 3, pp. 145–175, 2001.

[44] ——, "Building the gist of a scene: The role of global image features in recognition," Progress in brain research, vol. 155, pp. 23–36, 2006.

[45] N. Antropova, B. Q. Huynh, and M. L. Giger, "A deep feature fusion methodology for breast cancer diagnosis demonstrated on three imaging modality datasets," Medical physics, vol. 44, no. 10, pp. 5162–5171, 2017.

[46] Y. Ren, J. Yang, Q. Zhang, and Z. Guo, "Multi-Feature fusion with convolutional neural network for ship classification in optical Images," Applied Sciences, vol. 9, no. 20, p. 4209, 2019.

[47] S. L. Lee, M. R. Zare, and H. Muller, "Late fusion of deep learning and handcrafted visual features for biomedical image modality classification," IET image processing, vol. 13, no. 2, pp. 382–391, 2019.

[48] S. U. Amin, G. Muhammad, W. Abdul, M. Bencherif, and M. Alsulaiman, "Multi-cnn feature fusion for efficient eeg classification," in 2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW). IEEE, 2020, pp. 1–6.

[49] Y. Filali, H. E. Khoukhi, M. A. Sabri, and A. Aarab, "Efficient fusion of handcrafted and pre-trained cnns features to classify melanoma skin cancer," Multimedia Tools and Applications, vol. 79, no. 41, pp. 31 219–31 238, 2020.

[50] C.-J. Lin, C.-H. Lin, and S.-Y. Jeng, "Using feature fusion and parameter optimization of dual-input convolutional neural network for face gender recognition," Applied Sciences, vol. 10, no. 9, p. 3166, 2020.

[51] J.-A. Almaraz-Damian, V. Ponomaryov, S. Sadovnychiy, and H. Castillejos-Fernandez, "Melanoma and nevus skin lesion classification using handcraft and deep learning feature fusion via mutual information measures," Entropy, vol. 22, no. 4, p. 484, 2020.

DEEPAK THAKUR is pursuing Ph.D in Computer Science from Chitkara University, Punjab along with more than 6 years of teaching experience as an Assistant Professor in Computer Science and Engineering. His research interests include computer vision, malware classification, Android Security, and deep learning.

JAITEG SINGH is Ph.D in computer science and engineering with more than 15 years of experience in Research, Development, Training, Academics at institutes of Higher Technical Education. Areas of expertise are Software Engineering, Business Intelligence, Data, and Opinion Mining, Cartography, Curriculum Design, Pedagogical Innovation & Management. Areas of interest include Sustainable software engineering, Education Technology, Offline navigation systems, and Cloud computing.

TANYA GERA is pursuing Ph.D in Computer Science from Chitkara University, Punjab along with more than 6 years of teaching experience as an Assistant Professor in Computer Science and Engineering. Her research interests include Web Review Mining, Ransomware detection, Android Security, and machine learning.

BABAR SHAH is currently an Associate Professor with the College of Technological Innovation, Zayed University, Abu Dhabi Campus, United Arab Emirates. His professional services include but are not limited to Guest Editorships, University Services, the Workshops Chair, a Technical Program Committee Member, and a Reviewer of several reputed international journals and conferences. His research interests include WSN, WBAN, the IoT, churn prediction, security, real-time communication mobile P2P networks, and M-learning.

TAMER ABUHMED received the Ph.D. degree in information and telecommunication engineering from Inha University, in 2012. He is currently an Assistant Professor with the College of Computing, Sungkyunkwan University, South Korea. His research interests include biomedical applications, information security, network security, the Internet security, and machine learning and its application to medical, security, and privacy problems.

FARMAN ALI is an Assistant Professor in the Department of Software at Sejong University, South Korea. He received his B.S. degree in computer science from the University of Peshawar, Pakistan, in 2011, M.S. degree in computer science from Gyeongsang National University, South Korea, in 2015, and a Ph.D. degree in information and communication engineering from Inha University, South Korea, in 2018, where he worked as a Post-Doctoral Fellow at the UWB Wireless Communications Research Center from September 2018 to August 2019. His current research interests include sentiment analysis / opinion mining, information extraction, information retrieval, feature fusion, artificial intelligence in text mining, ontology-based recommendation systems, healthcare monitoring systems, deep learning-based data mining, fuzzy ontology, fuzzy logic, and type-2 fuzzy logic. He has registered over 4 patents and published more than 50 research articles in peer-reviewed international journals and conferences. He has been awarded with Outstanding Research Award (Excellence of Journal Publications-2017), and the President Choice of the Best Researcher Award during graduate program at Inha University.