applied sciences

MDPI

Article

Hybrid Malware Classification Method Using Segmentation-Based Fractal Texture Analysis and Deep Convolution Neural Network Features

Maryam Nisa [1], Jamal Hussain Shah [1], Shansa Kanwal [1], Mudassar Raza [1], Muhammad Attique Khan [2], Robertas Damaševičius [3] and Tomas Blažauskas [3,*]

1 Department of Computer Science, COMSATS University Islamabad, Wah Campus, Wah Cantt 47040, Pakistan; maryamnisa888@yahoo.com (M.N.); jamalhussainshah@gmail.com (J.H.S.); shansamalik10@gmail.com (S.K.); mudassarkazmi@yahoo.com (M.R.)

2 Department of Computer Science, HITEC University, Taxila 47080, Pakistan; attique@ciitwah.edu.pk

3 Department of Software Engineering, Kaunas University of Technology, 51368 Kaunas, Lithuania; robertas.damasevicius@ktu.lt

* Correspondence: tomas.blazauskas@ktu.lt

Received: 9 June 2020; Accepted: 17 July 2020; Published: 19 July 2020

Abstract: As the number of internet users increases so does the number of malicious attacks using malware. The detection of malicious code is becoming critical, and the existing approaches need to be improved. Here, we propose a feature fusion method to combine the features extracted from pre-trained AlexNet and Inception-v3 deep neural networks with features attained using segmentation-based fractal texture analysis (SFTA) of images representing the malware code. In this work, we use distinctive pre-trained models (AlexNet and Inception-V3) for feature extraction. The purpose of deep convolutional neural network (CNN) feature extraction from two models is to improve the malware classifier accuracy, because both models have characteristics and qualities to extract different features. This technique produces a fusion of features to build a multimodal representation of malicious code that can be used to classify the grayscale images, separating the malware into 25 malware classes. The features that are extracted from malware images are then classified using different variants of support vector machine (SVM), k-nearest neighbor (KNN), decision tree (DT), and other classifiers. To improve the classification results, we also adopted data augmentation based on affine image transforms. The presented method is evaluated on a Malimg malware image dataset, achieving an accuracy of 99.3%, which makes it the best among the competing approaches.

Keywords: malware; malicious code; convolutional neural network; deep features; feature fusion; transfer learning; image augmentation

1. Introduction

Malware programs are undesirable harmful threats that are intended to damage the security of a computer system. Malware detection has become an essential concern in the cybersecurity community since malware is capable of causing excessive loss and harm to computer security. Every day, a huge amount of malware is generated intentionally. A recent Symantec report in 2019 [1] demonstrated that malware is growing by 36% annually and the total samples of malware are estimated to be beyond 430 million. The rapid growth of malware causes an extensive threat in our daily life. Ransomware locks or encodes the files on a user's device and asks for a payment to restore them, which may lead to large productivity loses [2]. Data breaches due to malware activity often incur huge financial losses for major corporations [3]. Trojans and spyware are used in cyber espionage resulting in damage of geopolitical

and international relations [4]. Malware has become a primary concern for computer network [5] and smartphone users [6].

To counter these threats, deep learning, a technique based on artificial neural networks (ANN), can be employed successfully [7,8]. Deep learning with its multilayer architecture has a great ability to learn the features of the labeled and unlabeled data. However, when using a deep learning model, we need to train the model on a large data set every time, which takes a lot of time and computational resources. To overcome this learning difficulty, we can use pre-trained deep learning network architectures and extract features to build the classification model [9]. This motivates us to develop pre-trained network-based deep learning architecture for malware class identification and classification.

In this paper, we utilize binary files of malware as black-and-white (grayscale) images, which are generated using the byte code information and later used for malware classification. Visualizations of malware [10,11] in such form prove to be efficient in getting a complete view structure of images.

The conversion of textual data into image form is one of the major problems of malware detection [12]. In parallel, most of the malware detection methods adopt machine learning techniques by analyzing the different features within the malicious codes. These techniques generally experience two major problems: (1) extraction of robust and efficient features effectively for malware detection [13] and (2) data imbalance [14] preventing accurate malware detection and classification. Hence, keeping these problems in the mind, the ultimate challenge is to develop a malware identification model that can address the enormous variations in malware families.

In this paper, we propose a feature fusion-based solution to classify malware into different classes using features extracted from images as well as from pre-trained deep convolutional networks. The contributions of our proposed method are as follows:

- The conversion of malware binaries into the grayscale image.
- Data augmentation performed on malware images to overcome the data imbalance within the malware datasets for robust feature extraction to enhance the classifier performance.
- An optimized multimodal feature representation to combine the segmentation-based fractal texture analysis (SFTA) features [15] and deep convolutional neural network (DCNN) features into a single feature vector to obtain a robust malware classification model.

The other parts of the paper are prepared as follows: Section 2 discusses the related work, whereas Section 3 defines the detail of the proposed methodology. The results and analysis of experimental results and their significance are given in Section 4. Section 5 concludes the paper.

2. Related Work

In recent times, deep learning has confirmed its supremacy for many computer vision and machine learning applications like action recognition [16], gait recognition [17,18], object detection [19,20], and many more [21–23]. For malware detection and classification, different researchers have applied deep learning and image processing techniques to accomplish high accuracy because of their ground-breaking capacity to learn the best features.

For example, Cui et al. [12] presented a novel deep learning-based method for malware variant detection. They generated grayscale images from malicious code and effectively addressed the data imbalance problem by using bat algorithm. Then, a convolutional neural network (CNN) was used for identification and classification of malware images and the results show that the model achieved a good accuracy rate. Vinayakumar et al. [7] presented a more scalable and hybrid deep learning framework method for effective visual detection of malware with the same setup. Namavar Jahromi et al. [24] used an extreme learning machine with two hidden layers to detect malware threats in safety-critical systems. Zhu et al. [25] propose deep flow, a novel deep learning-based methodology for detecting malware straightforwardly from the information streams in the Android application. The results show that deep flow can accomplish a high discovery F1 score of 95.05%, outflanking customary deep learning based approaches, which uncovers the upside of deep learning procedure in malware

detection. Jeon and Moon [26] extracted the opcode sequences from a malware binary file and adopted a deep recurrent neural network (DRNN) as a classifier to detect malware. Sung et al. [27] used word embedding techniques to embed the malware codes into lower dimensionality feature space. Then, they applied a bi-directional long short-term memory (Bi-LSTM) network model to categorize the malware files by attack family. Gibert et al. [28] combined hand-engineered features and neural networks for malware classification. Each of the subnetworks can be trained separately, while the learned features are fused into a common representation.

To increase the classifier efficiency, researchers adopted deep learning as a tool for feature extraction to detect the malware code. Venkatraman et al. [29] presented a hybrid deep learning visualization approach and the result showed that the deep learning-based model effectively differentiates the behavioral pattern of different malware families. Zhong et al. [30] presented a technique which robustly handles the malware dataset complexities. This technique used a multi-level deep model, which organizes the tree structure of multiple deep models for enhancing the scalability of anonymous malware detection. Ye et al. [31] presented a heterogeneous deep model, which is composed of different layers of associative memory and a weighted auto encoder with multilayer restricted Boltzmann machines to identify malware. For unsupervised feature extraction, the deep learning model is forced to perform a training operation as a greedy layer-wise methodology followed by fine-tuning of a supervised parameter to efficiently detect the malware. Yuxin et al. [32] used a deep belief network (DBN) to train a multilayer generative model using the unlabeled data, which represents the better characteristics of employed data samples. DBNs are used as an auto encoder for feature extraction to detect malware. Vasan et al. [33] used handcrafted features as well as those of VGG16 and ResNet-50 CNNs to perform image-based malware classification. Čeponis and Goranin [34] analyzed the use of dual-flow deep learning methods, such as gated recurrent unit fully convolutional network (GRU-FCN) vs single-flow convolutional neural network (CNN) models for detection of malware signatures.

Over the last few years, the Android operating system (OS) has experienced tremendous popularity. However, this popularity comes at the expense of security, because it is an alluring target for malicious apps. Billah et al. [35] used deep learning as sequences for classification and proposed MalDozer, which is a family attribution and automatic Android malware detection model. MalDozer detects and learns the patterns of malicious and benign data from the actual dataset to expose Android malware. Pektaş and Acarman [36] used deep learning to recognize malware based on the application programming interface (API) call graphs transformed into a numeric feature set as a representation of the malware's execution paths. D'Angelo et al. [37] encoded the sequences of API calls invoked by apps as sparse image-like matrices (API images). Then, they used autoencoders to get the most informative features from these images, which were provided to an ANN-based classifier for malware detection. Naeem et al. [38] converted a raw Android file into a color image and then submitted it to a DCNN model, achieving 97.81% accuracy on a Leopard Mobile malware dataset and 98.47% accuracy on a Windows dataset. Vidal et al. [39] used genetic sequence alignment methods and statistical hypothesis testing for Android malware recognition, achieving an average hit rate of 98.61% on several public datasets.

The related works are summarized in Table 1. Here, the static analysis uses only static resources, which are available before the installation and execution of malicious applications [40].

Different from the earlier proposed methods, the proposed hybrid method based on feature fusion from pre-trained DCNN and SFTA multimodal feature representation is used for malware detection. The foremost objective of the proposed methodology is to extract the robust feature using malicious images to improve the classification accuracy after malware detection.

Table 1. Summary of some recent malware detection approaches.

Case Study	Year	Data Analysis	Dataset	Classification Approach	Accuracy
Malicious Code Localization [41]	2018	Static	Malware Dataset Benign Dataset Wild Dataset	Graph Kernels Support Vector Machine (SVM)	94%
Malware Variants Classified Behaviors [42]	2018	Dynamic	1220 malware samples	SVM, Decision Tree, Naïve Bayes	88.3%
Detection of Android Native Code Malware Variants [43]	2018	Static	Benign Dataset Malware Dataset	SVM	93.22%
Detecting and Classifying Android Malware [44]	2017	Static	Benign Dataset Malware Dataset	SVM	90%
Android Malware Detection System Based on Machine Learning [45]	2018	Hybrid	Benign Dataset Malware Dataset	SVM	95.2%
Malicious Code Variants Based on Deep Learning [12]	2018	Hybrid	Malware Dataset	SVM	94.5%
Image-Based malware classification [33]	2020	Static	Malware Dataset	Ensemble of convolutional neural networks	99.5%

3. Methodology

3.1. Outline of Methodology

The proposed methodology for malware classification has four major steps: (a) visualization of binary to grayscale image, (b) image augmentation, (c) feature extraction and fusion, and (d) feature selection and classification. The flow diagram of the proposed methodology is illustrated in Figure 1.

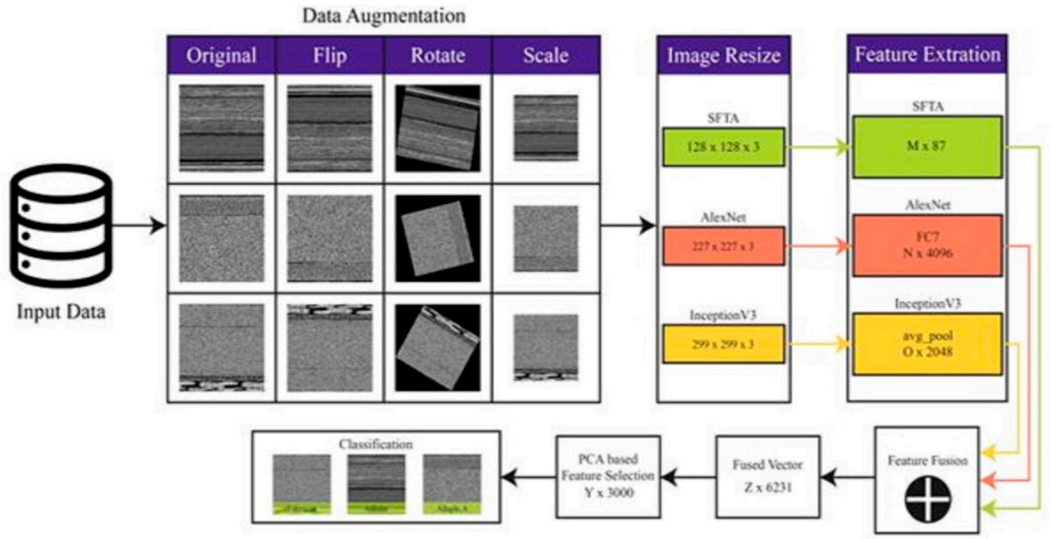

Figure 1. Flow diagram of the proposed malware classification method.

The proposed method uses DCNN architecture with a combination of SFTA (scale feature texture analyzer) features [15] for malware classification. Firstly, we convert the binary dataset files into grayscale images for accurate malware classification. To overcome the challenge of the data imbalance problem between different malware classes, we use a data augmentation method [46]. In the feature extraction step, the SFTA features are extracted from grayscale images and the pre-trained DCNN models of AlexNet [47] and Inception-v3 [48] are used for deep feature extraction. Both SFTA features and DCNN features are combined into a single feature vector using a serial-based feature fusion

method, and the most informative and robust features are selected using a principal component analysis (PCA)-based feature selection method for accurate malware classification. The complete details of each step are listed in further subsections below.

3.2. Visualization of Binary to a Grayscale Image

While dealing with malware, it is difficult to decide the type of attack by viewing the binary malware data. We convert the parallel malware executable file into 8-bit decimal value with a scope of 0–255, which is saved in a decimal number vector, representing a malware sample. We select the dimensions of the 2D matrix depending on the size of the malware binary file. At that point, we convert the grayscale image into three channel (red (R), green (G), blue (B)) images by duplicating the grayscale channels for three slices. A well-known change technique to change RGB (red, green, blue) to grayscale images is shown in Equation (1).

$$Gray_{image} = Red * 0.299 + Green * 0.587 + Blue * 0.114 \quad (1)$$

The process of visualization of binaries into a grayscale image is shown in Figure 2.

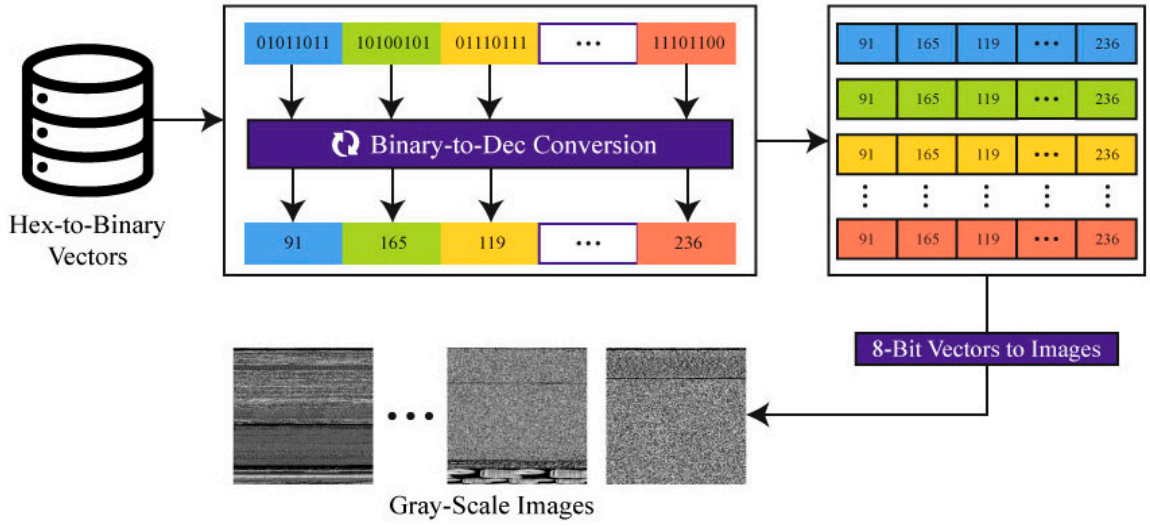

Figure 2. Conversion process of malware binaries into grayscale images for further processing.

The width of each gray scale malware image was 256 pixels, while the height relied upon the file size.

3.3. Image Augmentation

The imbalanced datasets are a communal problem in computer vision and harm classification problems. A deficiency of images in each class may cause under-fitting and over-fitting [10], which has a large influence on the performance of the DCNNs. Hence, to overcome this problem, a data augmentation method is proposed within the malware datasets to enhance the classifier performance. Image augmentation is a method that is used to increase the dataset that is commonly used to train the neural network [49]. In the augmentation phase, we generate new data from classes that have less population in the dataset [50]. This process overcomes the restrictive impact on data to avoid uneven representation. To escape over-fitting complications successfully, proper data augmentation schemes can be helpful and improve the strength of the model.

Let D_1 be a given malware dataset, $D_1 = \{a_i \ldots\ldots\ldots a_n\}$, where a_n represents the nth image sample in the dataset. Assume that a_n has N pixels. The pixel coordinate matrix A_k for a_j is given below in Equation (2).

$$A_k = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ . & . & . \\ x_N & y_N & 1 \end{bmatrix} \tag{2}$$

here each row shows similar coordinates of one pixel. For data augmentation on image a_n, we apply an affine transform matrix, p, on the coordinate matrix, M_k, and obtain a transformed coordinate matrix, M_t^k, for the image. The operation is given in Equation (3) as follows:

$$M_t^k = p * M_k, \tag{3}$$

where each row of M_t^k is the transformed coordinate for one pixel. The transform matrix operations are also shown in Table 2.

Table 2. Affine transformation matrices for image augmentation.

Operations	Flipping	Scaling	Rotation
Matrix transform	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$

There are different approaches to decide the affine transformation matrix M. We utilize three sorts of arbitrary irritations for creating new augmentation data as follows. Flipping operation flips the image along the horizontal dimension. Rotation operation pivots the image with a point inspected from 0° to 180°. Scaling operation moves the image in both the x and y dimension of the image. The respective affine transform matrices are shown in Table 2, where β shows the angle of rotation T_x, and T_y shows the balances on the arrange hub.

Let a_k be a given image the augmented details denote as A_k, and the dataset is denoted as D_1. The augmentation process is formulated in the following equations:

$$A_k = \{T_1(a_k), T_2(a_k), T_3(a_k)\}, \tag{4}$$

$$D_1 = \{A_1 \ldots\ldots\ldots A_k\} = \cup_{k=1}^{k} \cup_{j=1}^{3} \; T_i(a_k), \tag{5}$$

where T_1 is flipping, T_2 is rotation, and T_3 is scaling operations.

The expanded dataset, D_1, alongside the compared class names, is then employed to train a deep CNN with the end goal of malware recognition and classification. It ought to be noticed that we misuse flips, rotation, and scaling as the data augmentation tasks since they do not modify the texture topologies in malware classes, as shown in Figure 3.

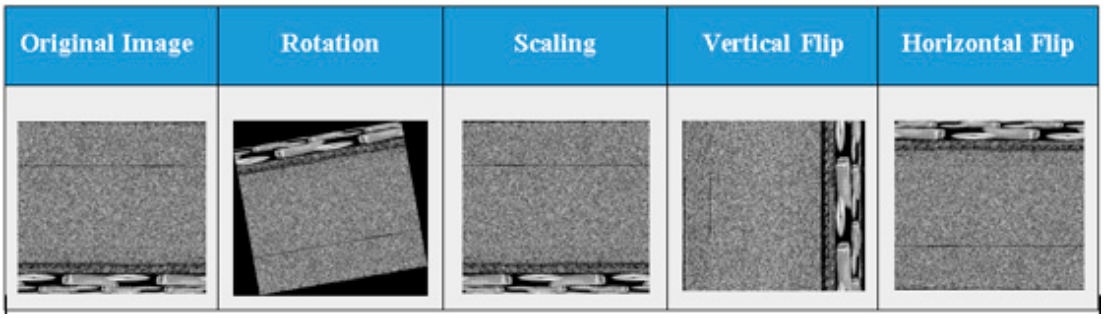

Figure 3. Data augmentation process. Images are generated after applying augmentation techniques on the Swizzor.gen! malware family from the Malimg dataset [51].

The augmentation process for example flips, rotates, and scales up a decent variety of training data and improves the classification accuracy of deep CNNs. The data augmentation method in Algorithm 1 describes how the malware images are rotated and flipped to deal with the problem of class imbalance, as given in Algorithm 1.

Algorithm 1. Malware Image Augmentation

Input: Malware images $Y = \{(y_i), i = 1, 2, 3 \ldots\ldots\ldots, X_y\}$, image flipping (Vertical and Horizontal) and rotation $MR = mr_k, k = 1, 2, \ldots\ldots\ldots, mi_{xi}$ where mr_k is the rotation angle.
Output: Rotation of malware image *RI* and flipped malware image *FI*.

```
for each Image y_i ∈ Y do
    for k = 1 to
        Angle of rotation mr_k to X_y
    end for
    for MI = {Horizontal, Vertical}
        Flipping
    end for
end for
return RI and FI
```

3.4. Feature Extraction

Feature extraction is an important step in Malware classification. When the size of the data is large and difficult to manage, then the data is transformed into a reduced set of feature representations. A feature holds data about the size, texture, color, and shape of an image. Here, we used local features, global features, and texture features. The proposed framework adopts two stages for feature extraction, which are performed using the equivalent stream. In the first stage, from grayscale images, SFTA is used for texture feature extraction because of its robustness and low computation cost. In the second stage, the pre-trained CNNs, like Alexnet and Inception-v3, are used to extract deep features in depth to obtain a robust classification model. Each feature extraction method is discussed in the following sections.

3.4.1. Texture Feature (SFTA)

Segmentation-based fractal texture analysis (SFTA) is a texture-based feature acquisition method [15]. The texture element is the most noteworthy trait of an image, which is utilized to identify and characterize the malware images and discover the similitudes between images of various malware families. SFTA is utilized for texture feature extraction because of its robustness and low computation cost. The SFTA extraction method can be partitioned into two parts, which are discussed below.

First, the input grayscale image $i(v, u)$ is broken down into a binary image. We utilize two-threshold binary deterioration (TTBD) for binary decomposition. Binary images are formulated in Equation (6).

$$i_b(v, u) = \left\{ \begin{array}{c} \&_0^1 \; if \; t_{low} \leq 1(v, u) \leq t_{up} \\ \text{otherwise} \end{array} \right\}, \tag{6}$$

where t_{low} is the lower threshold and t_{up} represents the upper threshold value.

After applying TTBD to the gray level input image, the SFTA feature vector is developed as the fractal dimension size, mean gray level, and, furthermore, the SFTA feature vector. The fractal estimations are utilized to portray the complexity of malware image structures fragmented in the input

image. The fractal dimension D_i is determined by using the boundary around an image. The boundary image, $\Delta(v,u)$, is computed as follows:

$$\left\{ \begin{array}{l} 1 \text{ if } \exists(v',u') \in N_{I=8}[(v,u)]: \\ \quad I_b(v',u') = 0 \wedge \\ \quad I_b(v',u') = 1 \\ 0 \text{ otherwise} \end{array} \right\}, \tag{7}$$

where $N_{I=8}[(v,u)]$ is the connected 8 pixels to (v,u), $\Delta(v,u)$ in the event that the pixel at (v,u) in the binary image $I_b(v',u')$ has the value of 1 and having in any event one neighboring pixel with esteem 0. Otherwise, the value is 0. $\Delta(x,y)$ takes the value of 0.

Consequently, one can understand that the comings about outskirts are one pixel wide. The gray level mean size (pixel tally) supplements the data extricated from every binary image without essentially expanding the calculation time. Algorithm 2 and Figure 4 show the SFTA feature extraction process.

Algorithm 2. SFTA Feature Extraction Algorithm

Require: *Grayscale image* $i(v,u)$ and two thresholds t_{low} and t_{up}.
Ensure: SFTA feature vector $s_{1\times m}$.
1: $T_a \leftarrow$ **MultiLevelOtsus** $i\left(t_{low}, t_{up}\right)$
2: $T_b \leftarrow \left\{\{t_i, t_{i+1}\} : t_i, t_{i+1} \in T_a, i \in \left|1\ldots\right|T||-1|\right\}$
3: $T_c \leftarrow \left\{\{t_c, t_l\} : t_i \in T_a, i \in \left|1\ldots\right|T|\right\}$
4: $i \leftarrow 0$
5: **for** $\{\{t_l, t_u\} : \{t_l, t_u\} \in T_b \cup T_c\}$ do
6: $I_b \leftarrow$ **Two Thresholds** $\left(i, t_{low}, t_{up}\right)$
7: $\Delta(v,u) \leftarrow$ **Find Borders** I_b
8: $s_{1\times m}[i] \leftarrow$ **Box Counting** (Δ)
9: $s_{1\times m}[i+1] \leftarrow$ **MeanGrayLevel** (i, i_b)
10: $s_{1\times m}[i+2] \leftarrow$ **Pixel Count** I_b
11: $i \leftarrow i+3$
12: **end for**
13: **return** $s_{1\times m}$

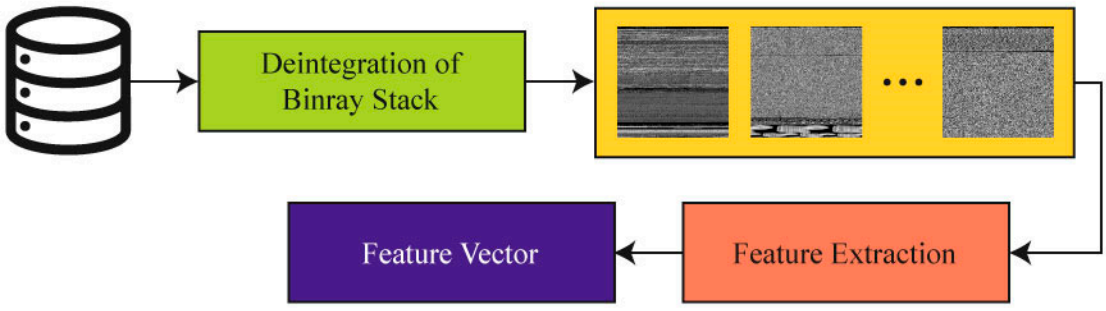

Figure 4. Segmentation-based fractal texture analysis (SFTA) algorithm for the extraction of texture features from malware images.

3.4.2. Deep Convolution Neural Network (DCNN)

Here we use a deep convolutional neural network (DCNN) model for malware classification and recognition. The network architecture has a multilayer structure comprising of convolutional layers, pooling layers, and fully connected layers. Every neuron is processed by a spot item between their little areas and weight, which are associated with the information volume. From that point, activation is performed utilizing the ReLu layer. The ReLu layer does not change the size of an image. Thereafter, the pooling layer is performed to lessen the noise impacts in the separated features. Lastly, significant

level features are determined by an associated fully convolutional (FC) layer. The architecture of the neural network is shown in Figure 5.

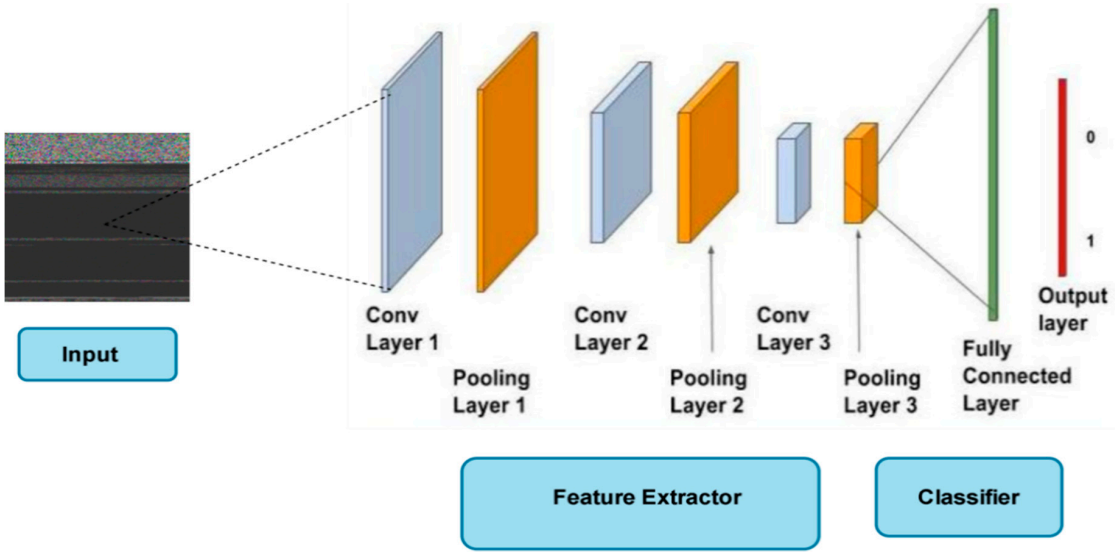

Figure 5. Framework of deep convolutional neural network (CNN) for feature extraction from malware images.

In this paper we utilize two pre-trained CNN models (AlexNet [47] and Inception-V3 [48]) for feature extraction. These models link a convolution layer, pooling layer, normalization layer, ReLu layer, and FC layer. Deep convolution layer local feature extraction from an image is expressed in the following equations.

Let us denote the M_i^N output layer N, and b_i^N the base value, then, the filter for the jth feature map is $\psi_{i,j}^N$, and h_j denotes the $N-1$ th output layer. Then, the deep convolutional layers have the formulation as stated in Equation (8).

$$M_i^N = b_i^N + \sum_{j=1}^{m_1\ (N-1)} \psi_{i,j}^N \times h_j^{N-1} \tag{8}$$

The pooling layer extricates the most extreme returns from the lower convolutional layer with the goal of removing unnecessary features. The maximum pooling settles the issue of over-fitting and a 2 × 2 matrix is performed on the extricated framework. Max pooling is depicted in Equations (9)–(11).

$$f_1^K = f_1^{k-1}, \tag{9}$$

$$f_2^k = \frac{f_2^{k-1} - F(k)}{S^k} + 1, \tag{10}$$

$$f_3^k = \frac{f_3^{k-1} - F(k)}{S^k} + 1, \tag{11}$$

Here, S^k denotes the stride and f_1^k, f_2^k, and f_3^k are 2×2 and 3×3 filters for feature maps. The ReLu and FC (fully connected) layers are defined as:

$$R_i^l = \max\left(h, h_i^{l-1}\right), \tag{12}$$

$$FC_i^l = f\left(z_i^l\right) with\ z_i^l = \sum_{q=1}^{m_1\ (l-1)} \sum_{r=1}^{m_2\ (l-1)} \sum_{s=1}^{m_3\ (l-1)} x_{i,j,r,s}^l \left(FC_i^{l-1}\right)_{r,s} \tag{13}$$

Here, R_i^l represents the ReLu layer and FC_i^l represents the FC layer. The FC layers follow the convolution and pooling layers. The FC layer is like a convolution layer, and the greater part of the analysts perform activation on the FC layer for deep features extraction.

3.4.3. DCNN Feature Extraction and Fusion

Here we use two distinctive pre-trained models (AlexNet, Inception-V3) for feature extraction. These models have different structures, but they have been trained with the same dataset of malware.

AlexNet [47] contains five progressive convolutional layers, Conv1 · Conv5, succeeded by three max-pooling layers, and three FC layers, FC7 with a last softmax classifier. The AlexNet CNN has been trained on 1.2 million images of 1000 classes, with around 60 million network parameters. The network only accepts RGB images with a dimension size of $224 \times 224 \times 3$.

The Inception-V3 [48] model is composed of well-structured convolution modules that can both produce preferential features and decrease the quantity of the parameters. Every Inception module is made of a few convolutional layers and pooling layers in equal numbers. Convolutional layers, for example, 3×3, 1×3, 3×1, and 1×1 layers, are utilized in the Inception modules to diminish the quantity of parameters.

The purpose of DCNN feature extraction from two network models is to improve the classifier accuracy. Since both models have characteristics and qualities of different extract features. Therefore, three kinds of features are extracted and used: AlexNet features, Inception-V3 features and texture (SFTA) features. We extract $M \times 87$ features from SFTA and deep features by performing an activation function on the FC7 layer and applying max pooling to expel the noise. The FC7 layer extracted $N \times 4096$ features, and $O \times 2048$ features are extracted through the average pooling layer, where M, N, and O represent the number of images passed to the model.

To get efficient results, we perform serial based fusion to fuse the deep features and texture feature SFTA. Three feature vectors are defined as follows. Let $1 \times M$ represent the SFTA feature vector, $1 \times N$ represent the AlexNet feature vector, and $1 \times O$ represent the Inception-V3 feature vector, then, Equations (14)–(16) are formulated for the features vector:

$$f_{s \times m} = \{s_{1\times1}, s_{1\times2}, s_{1\times3}, s_{1\times4}, \ldots\ldots, s_{1\times m}\}, \tag{14}$$

$$f_{a \times n} = \{a_{1\times1}, a_{1\times2}, a_{1\times3}, a_{1\times4}, \ldots\ldots, a_{1\times n}\}, \tag{15}$$

$$f_{i \times o} = \{i_{1\times1}, i_{1\times2}, i_{1\times3}, i_{1\times4}, \ldots\ldots\ldots, i_{1\times o}\}, \tag{16}$$

Then, the resulting feature vectors from SFTA, AlexNet and Inception-v3 are concatenated. These feature vectors are shown below:

$$Fused_{fv} = \sum_{i=1}^{3} \{f_{s\times m}, f_{a\times n}, f_{i\times o}\}, \tag{17}$$

$$\sum (f_s, f_a, f_i) = \begin{pmatrix} s_{1\times1}, s_{1\times2}, s_{1\times3}, s_{1\times4}, \ldots\ldots, s_{1\times m}, \\ a_{1\times1}, a_{1\times2}, a_{1\times3}, a_{1\times4}, \ldots\ldots, a_{1\times n}, \\ i_{1\times1}, i_{1\times2}, i_{1\times3}, i_{1\times4}, \ldots\ldots, i_{1\times n} \end{pmatrix}, \tag{18}$$

$$f_{Resu_{1\times q}} = \left\{p_{1\times1}, p_{1\times2}, p_{1\times3}, p_{1\times4}, \ldots\ldots, p_{1\times q}\right\}, \tag{19}$$

where $f_{Resu_{1\times q}}$ is the resultant feature vector, and $1 \times q$ represents the dimension $q = (m, n, o)$.

3.5. Feature Selection

The inspiration behind the employment of the feature selection is to select the most prominent features to refine the accuracy and eliminate the unnecessary features to make the system fast in terms of execution time and increase the accuracy rate. In this paper, from the fused feature vector (FV),

principal component analysis (PCA) is used for optimal feature selection. PCA has the following steps: (a) calculating the mean of a fused feature vector, (b) subtracting the mean from each feature, (c) calculating the covariance matrix, and (d) calculating the eigenvalues and eigenvector from covariance matrix. Finally, PCA returns a score value and a principal components score.

The first process before applying PCA is normalizing the FV so that it works adequately. It can be done by subtracting the relevant means from the corresponding column in the FV. Let us have FV values denoted by x, and N is the total number of extracted feature values in FV:

$$x_0 = \frac{1}{N} \sum_{i=0}^{N} x_i, \tag{20}$$

$$K_0(x_0) = \sum_{i=0}^{n} \left| (x_0 - x_1) \right| 2, \tag{21}$$

where K_0 will be the normalized FV that will be used further for finding the covariance matrix.

$$conv(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - x')(Y_i - y') \tag{22}$$

The covariance of $conv(x, y)$ matrix calculates the eigenvectors with their corresponding eigenvalues. The selected optimal features from both models after applying PCA are fused using a parallel-based method and a final fused vector is obtained.

4. Results and Analysis

4.1. Dataset

The Malimg [51] dataset (available for download from https://www.dropbox.com/s/ep8qjakfwh1rzk4/malimg_dataset.zip) consists of malware images that were processed from malware binaries, and we trained the deep learning models to classify each malware family. The dataset comprised of 25 malware families and has 9348 grayscale images. Each family of the Malimg dataset contains a different number of samples, so the dataset is imbalanced (see Figure 6). The dataset has an unfair distribution of classes, for example, 2949 images represent the Allaple malware family, while only 80 images are present in the Skintrim family. The image augmentation techniques are used to stabilize the number of training samples to recover the superiority of data trials. By applying the data augmentation technique, the data imbalance problem is resolved, and each family of malware contains 1000 malware samples. Hence, the number of total images becomes 25,000.

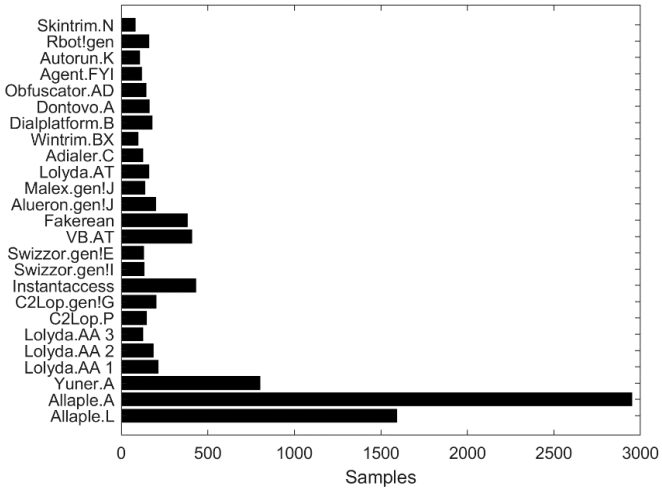

Figure 6. Distribution of samples in the Malimg dataset [51].

4.2. Settings

For classification, we used a support vector machine (SVM) with linear, quadratic, polynomial (cubic), and Gaussian kernels, k-nearest neighbor (KNN) with cosine similarity as distance metrics (cosine KNN), KNN with the count of neighbors set to 1 (fine KNN), KNN with the count of neighbors set to 10 (medium KNN), KNN with the count of neighbors set to 100 (coarse KNN), KNN with distance weighting (weighted KNN), decision tree (DT), DT with the maximum count of splits set to four (fine DT), 20 (medium DT), and 100 (coarse DT), boosted DT, cosine DT, Adaboost classifier, and linear discriminate (LD) classifiers.

Each classifier performance is evaluated using recall, error rate (ER), and area under the curve (AUC) and accuracy metrics (Equations (23)–(25)). All experiments were done with MATLAB 2018b (Math Works, Inc., Natick, MA, USA) having machine specifications of 1.00 GHz CPU with 8 GB RAM.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{23}$$

$$\text{ER} = \frac{FP + FN}{TP + TN + FP + FN} \tag{24}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{25}$$

4.3. Results

We use the multi-representation of various features and selection methods incorporated to obtain diverse classifier results to verify the efficiency of the proposed method. The results are generated into three groups: (a) selection of features using traditional method (SFTA), (b) selection of features using traditional approach (SFTA) and pre-trained network (AlexNet, Inception-v3), and (c) deep CNN and SFTA feature fusion lengthwise with PCA selection method. All these experiments are employed on the malware dataset by applying PCA and the planned features collection method.

We claim that we can build a high level of accuracy in recognizing malware from trusted samples deep learning with features derived from the pre-trained network and handcrafted method. These features are continuous and allow us to be more flexible with the classification. We use these continuous data and develop a novel classification method using a pre-trained network and handcrafted method to reduce over-fitting during training. We compare our method to a set of machine classification methods that have been applied in previous research and demonstrate an increase of classification accuracy using our method and an unseen dataset over the range of other machine classification methods that have been applied in previous research.

Four experiments are performed using the Malimg balanced and imbalanced dataset to check the effectiveness of results, and the results are compared with the existing methods.

4.3.1. Experiment 1: With Imbalanced Data and SFTA Features without Feature Fusion and Feature Optimization

Experiment 1 is performed using the features obtained by applying SFTA on the Malimg dataset [51] and using the state-of-the-art classifiers. The imbalanced dataset has large different sized images in 25 classes. This shows the problem of over-fitting and has a negative influence on the results. The results based on imbalanced data and without feature fusion and feature optimization using SFTA are depicted in Table 3. The best accuracy is achieved using the quadratic SVM.

Table 3. Classification results using imbalanced dataset with SFTA features. Best values are shown in bold.

Classifier	Recall (%)	Accuracy (%)	AUC (%)	ER (%)
Cubic Support Vector Machine (SVM)	94.0	92.1	90.0	7.9
Decision Tree	36.0	72.4	81.0	27.6
Weighted k-Nearest Neighbor (KNN)	86.0	87.2	75.0	12.8
Fine Gaussian	49.0	81.4	**93.1**	18.6
Fine KNN	95.0	93.7	92.0	6.3
Boosted Decision Tree	85.9	85.2	92.0	14.5
Adaboost	91.0	90.2	89.0	9.8
Linear Discriminate	76.0	74.4	78.0	25.6
Quadratic SVM	**96.0**	**95.2**	90.0	**4.8**

The quadratic SVM on an imbalanced dataset obtained a maximum classification accuracy of 95.2%, recall of 69.0%, AUC of 90.0%, and ER of 4.8% on 10-fold cross-validation. Quadratic SVM obtained minimum ER and maximum AUC as compared to other classifiers such as weighted KNN, linear discriminate and some more in Table 3. The recall, ER, accuracy and AUC values of certain classifiers are also plotted in the graph shown in Figure 7.

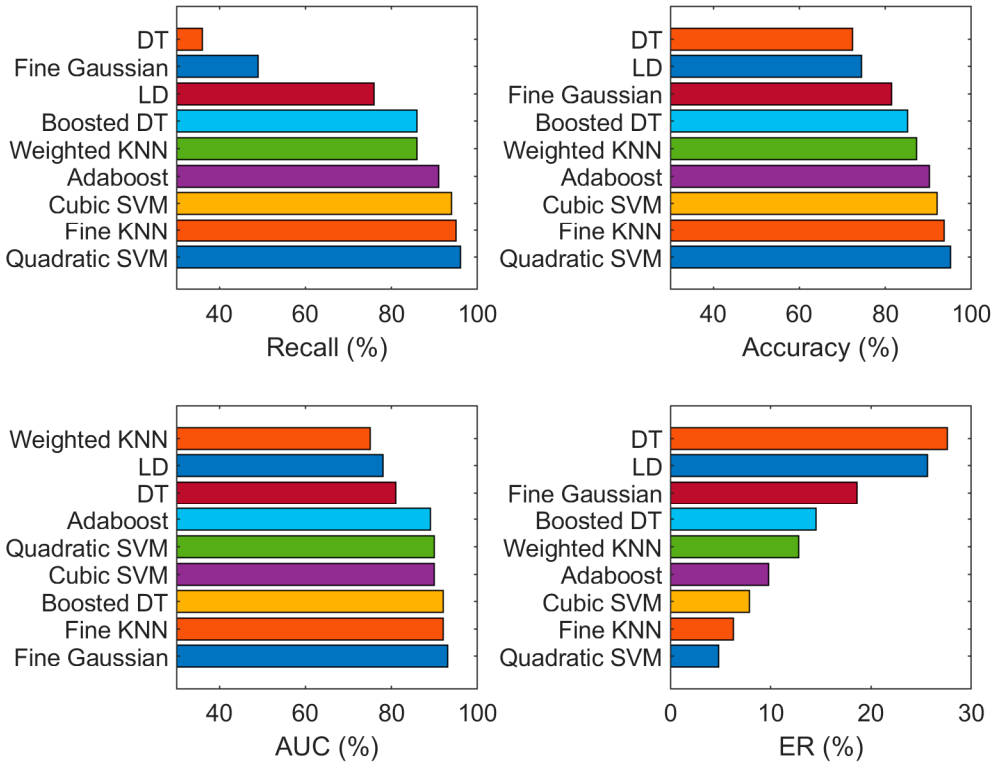

Figure 7. Classification results (recall, accuracy, AUC and ER) using SFTA features on the original (imbalanced) dataset.

4.3.2. Experiment 2: With Balanced (Augmented) Data and SFTA Features

Experiment 2 was performed on balanced (augmented) data, using the SFTA features but without any feature fusion and optimization. The classification results are depicted in Table 4.

Table 4. Classification results using balanced data and SFTA features. Best values are shown in bold.

Classifier	Recall (%)	Accuracy (%)	AUC (%)	ER (%)
Gaussian SVM	**98.0**	98.5	**100.0**	1.5
Decision Tree	72.0	76.8	99.0	23.2
Weighted KNN	95.0	96.3	99.0	3.7
Fine Gaussian	94.0	95.7	**100.0**	4.3
Boosted Decision Tree	98.0	98.4	**100.0**	1.6
Adaboost	96.0	95.2	99.0	4.8
Linear Discriminate	97.6	**99.3**	60.0	**0.7**
Cubic SVM	**98.0**	98.2	**100.0**	1.8

The linear discriminant (LD) classifier used on a balanced dataset obtained a maximum classification accuracy of 99.3%, recall of 97.60%, AUC of 60.0%, and ER of 0.7% on 10-fold cross-validation. In the balanced data, accuracy is improved, and ER is smaller as compared to the imbalanced dataset. The image augmentation to obtain the balanced dataset has a positive influence on the results and solves the problem of overfitting, so accuracy is improved. The recall, ER, accuracy and AUC values of classifiers are also plotted in the graph shown in Figure 8.

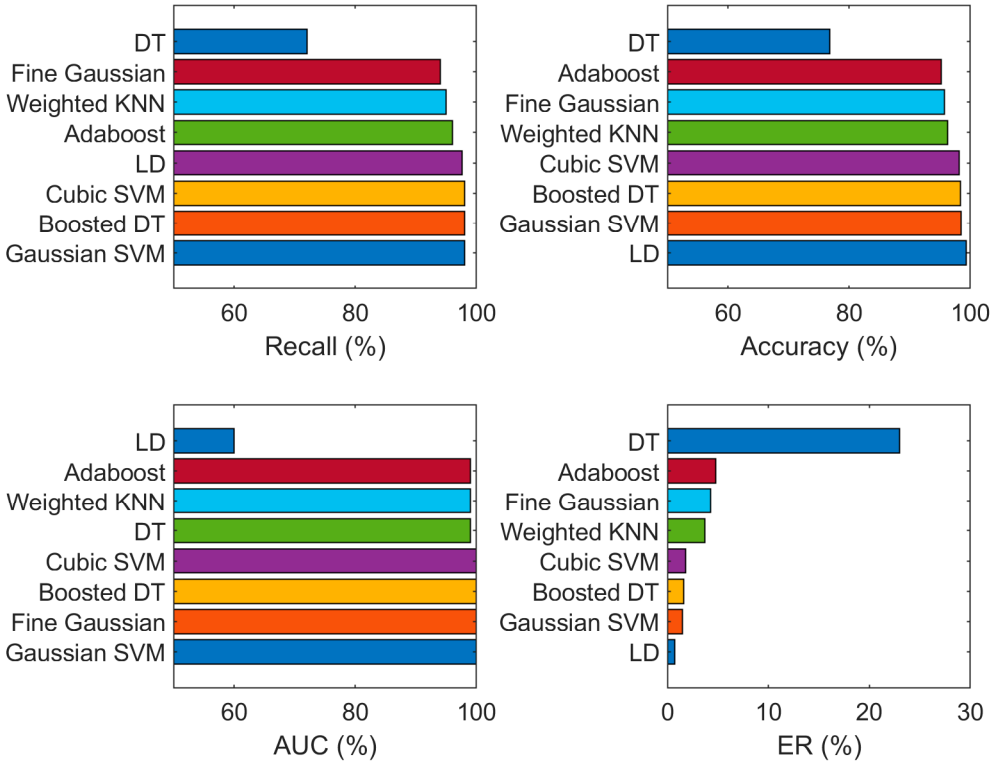

Figure 8. Classification results (recall, accuracy, AUC, and ER) using SFTA features on the balanced (augmented) dataset.

4.3.3. Experiment 3: With Imbalanced Data and Using Featured Obtained from a Pre-Trained Network Model

Two pre-trained neural networks (AlexNet and Inception-v3) are used as deep feature extractors, and the classification results are presented in Table 5.

Table 5. Imbalanced data without feature fusion and optimization results on pre-trained network. Best values are shown in bold.

Classifier	Recall (%)	Accuracy (%)	AUC (%)	ER (%)
Linear SVM	96.0	97.5	85.0	2.5
Fine Decision Tree	96.0	97.4	85.0	2.6
Weighted KNN	66.0	44.6	60.0	55.4
Fine KNN	67.0	54.8	63.0	45.2
Cosine KNN	97.0	**98.7**	88.0	**1.3**
Boosted Decision Tree	78.0	68.9	72.0	31.1
Quadratic SVM	96.0	97.5	85.0	2.5
Linear Discriminate	96.0	98.1	**88.0**	1.9
Cosine Decision Tree	72.0	58.6	63.0	41.8
Medium Decision Tree	69.0	51.4	60.0	48.6
Cubic SVM	**97.0**	98.3	**88.0**	1.7

The quadratic SVM used an on imbalanced dataset performed well and obtained a maximum classification accuracy of 98.3%, recall of 97.0%, AUC of 88.0%, and ER of 1.7% on 10-fold cross-validation. The quadratic SVM obtained minimum ER and maximum AUC as compared to other classifiers such as weighted KNN, linear discriminate, and some more in Table 5. The recall, ER, accuracy and AUC of certain classifiers are also plotted in the graph shown in Figure 9.

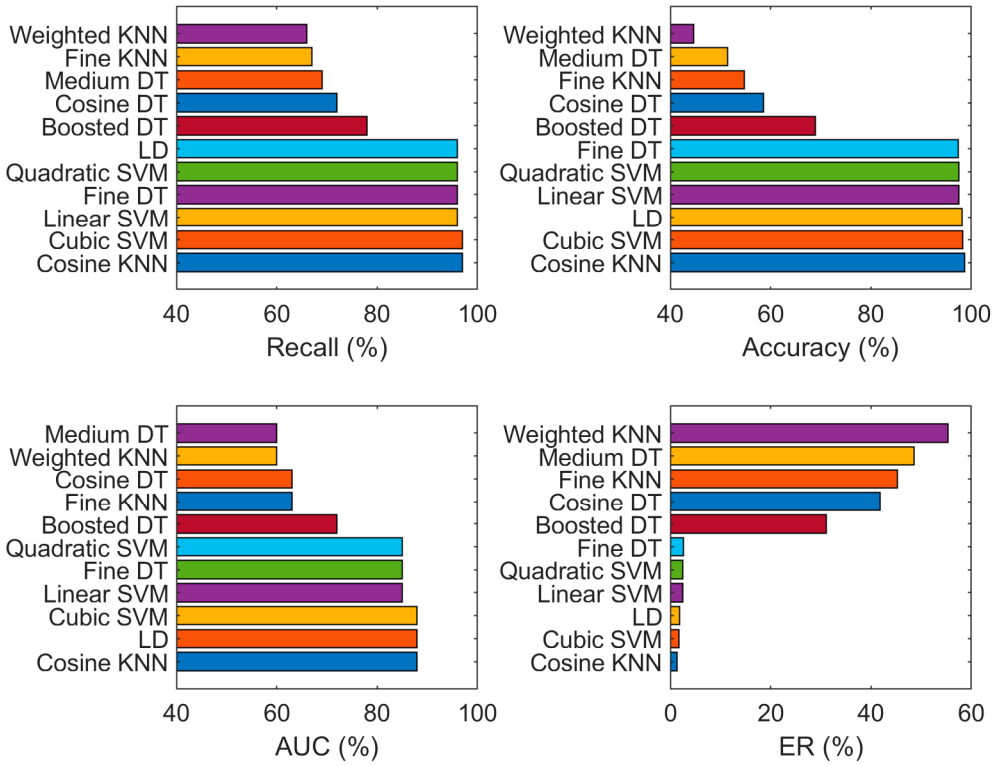

Figure 9. Classification results (recall, accuracy, AUC and ER) using a fusion of SFTA and pre-trained network features on the original (imbalanced) dataset.

4.3.4. Experiment 4: With Balanced Data and Fusion of SFTA and Pre-Trained Network Features

Experiment 4 aims to confirm the usefulness of a balanced dataset on a DCNN. For a DCNN, all malware balanced dataset images are resized to $227 \times 227 \times 3$. This experiment takes less training time as compared to the above experiments. Two pre-trained networks, AlexNet and Inception-V3, are used as deep feature extractors. The feature fusion of the pre-trained network and handcrafted method (SFTA) results are revealed in Table 6, and their visualization is shown in Figure 10.

Table 6. Balanced (augmented) data with feature fusion of SFTA and a deep convolutional neural network (DCNN). Best values are shown in bold.

Classifier	Recall (%)	Accuracy (%)	AUC (%)	ER (%)
Linear SVM	98.0	98.5	**100.0**	1.5
Fine Decision Tree	77.0	82.8	97.0	17.2
Weighted KNN	95.0	96.3	99.0	4.3
Medium KNN	93.0	94.7	99.0	6.7
Fine KNN	97.0	97.7	98.0	3.7
Cosine KNN	96.0	96.9	99	4.9
Boosted Decision Tree	97.0	72.8	99.0	27.2
Quadratic SVM	95.0	99.1	**100.0**	**1.0**
Linear Discriminate	97.0	97.6	99.0	3.6
Cosine Decision Tree	21.0	33.5	37.0	66.5
Medium Decision Tree	48.0	66.8	84.0	33.2
Cubic SVM	**99.0**	**99.3**	**100.0**	1.3

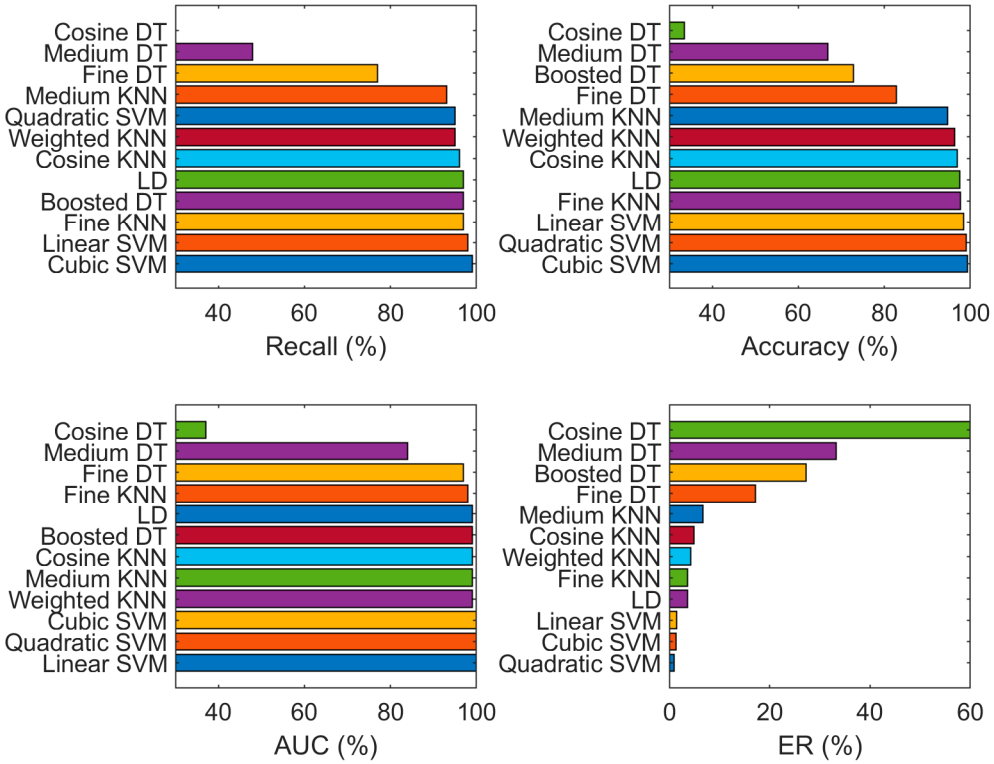

Figure 10. Classification results (recall, accuracy, AUC and ER) using a fusion of SFTA and pre-trained network features on the balanced (augmented) dataset.

Table 6 shows a maximum accuracy level of 99.3%, recall of 99.0%, AUC of 100.0%, and ER of 1.3% on cubic SVM, as compared to some more classifiers depicted in Table 6, which shows the effectiveness of balanced data and the robustness of the proposed technique. The pre-trained network works well with balanced datasets and obtained maximum accuracy on certain classifiers. The recall, accuracy, and AUC values of the balanced dataset are higher than for the imbalanced dataset.

4.4. Statistical Analysis

To assess the statistical significance of the results, we adopted the non-parametric Friedman test for the compared methods on all cross-validation folds. As the classification results are not Gaussian distributed, the Friedman test was applied to detect differences in the classification results of various methods across multiple runs. The results of the Friedman test regarding accuracy (see Figures 11–14)

show that the differences between the methods are statistically significant ($p < 0.01$). Critical difference (CD) shows the smallest difference in mean ranks, where the difference is not statistically significant.

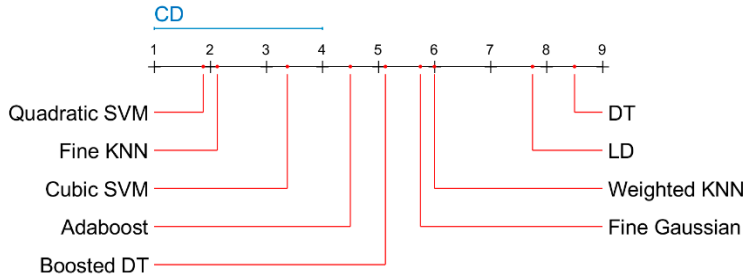

Figure 11. Critical difference diagram of the Nemenyi test results for malware classification using SFTA features (imbalanced dataset). CD is critical difference.

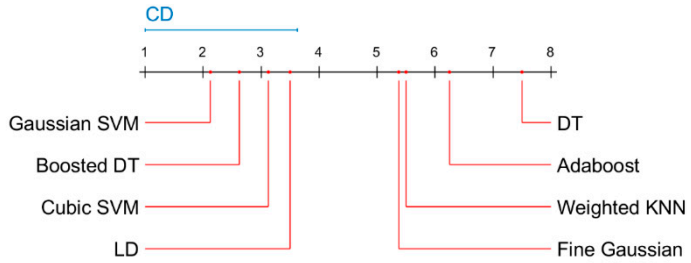

Figure 12. Critical difference diagram of the Nemenyi test results for malware classification using SFTA and features (balanced dataset). CD is critical difference.

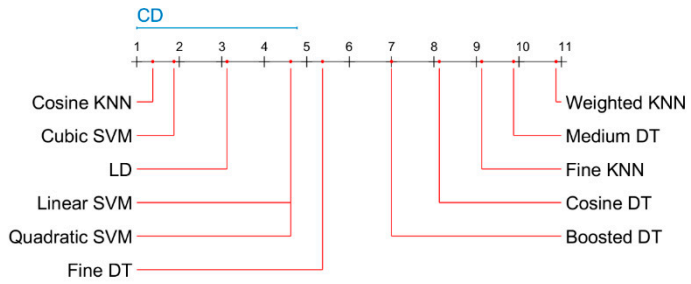

Figure 13. Critical difference diagram of the Nemenyi test results for malware classification using a fusion of SFTA and pre-trained neural network features (imbalanced dataset). CD is critical difference.

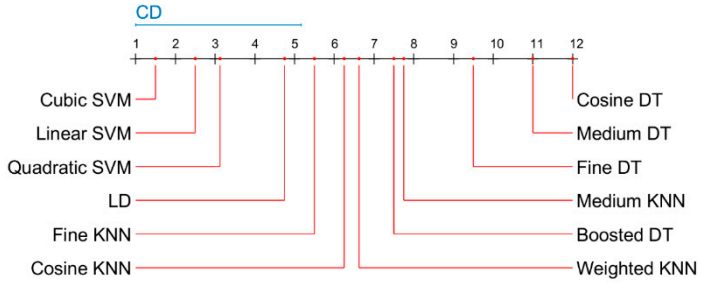

Figure 14. Critical difference diagram of the Nemenyi test results for malware classification using a fusion of SFTA and pre-trained neural network features (balanced dataset). CD is critical difference.

The confusion matrix for the best classifier (cubic SVM with a fusion of SFTA and pre-trained neural network features applied on the balanced dataset) is presented in Figure 15. We can see only a few misclassifications mostly between the two most numerous classes.

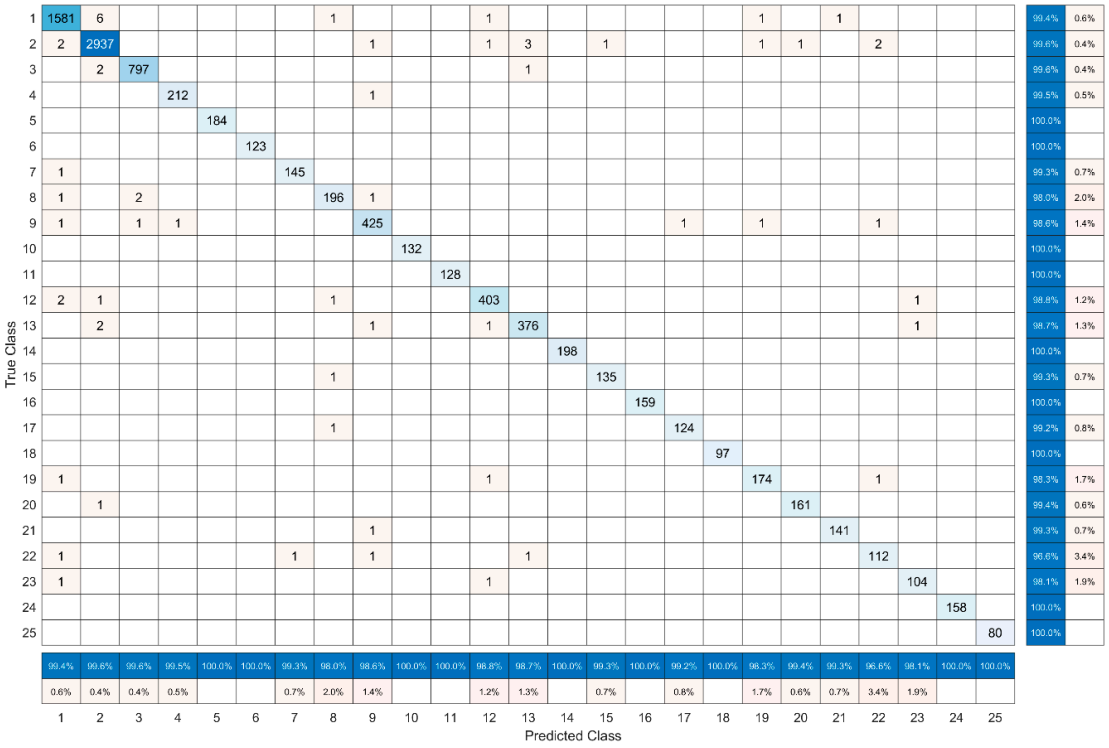

Figure 15. Confusion matrix of the best classifier (cubic SVM with a fusion of SFTA and neural network features).

The best classification results (based on accuracy) for all experiments are summarized in Figure 16. Note that image augmentation allowed a statistically significant improvement of results to be achieved for the SFTA features (by 4.1%, $p < 0.05$, using non-parametric Wilcoxon rank-sum test). However, for the fused feature set, the improvement was not statistically significant. On the other hand, using feature fusion on the original (imbalanced) dataset also allowed a statistically significant improvement to be achieved (by 3.5%, $p < 0.05$).

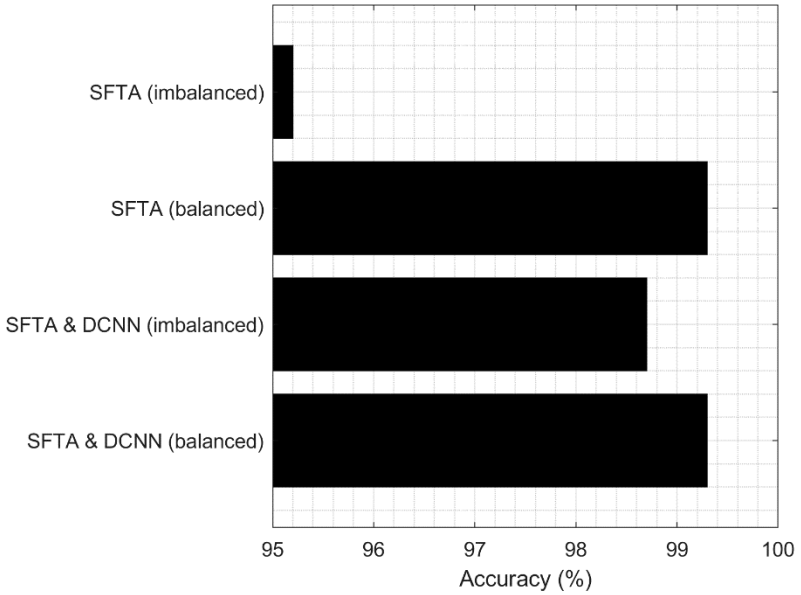

Figure 16. Comparison of best accuracy for malware classification for different feature sets (SFTA features vs a fusion of SFTA and neural network features) and imbalanced/balanced datasets.

4.5. Comparison of the Results of the Proposed Technique with Other Existing Methods

The results on the Malimg dataset are compared with the results of other authors using the same dataset in Table 7. Note that our method allowed the best results to be achieved on this dataset.

Table 7. Comparison with existing state-of-the-art algorithms.

Author	Year	Features	Techniques	Accuracy (%)
Nataraj et al. [51]	2011	GIST feature	Nearest Neighbor	97.18
Anderson et al. [52]	2012	Gaussian kernel features	SVM	98.0
Dahl et al. [53]	2013	Sparse binary features	Neural networks and logistic regression	86.0
Zhang et al. [54]	2014	Graph-based feature	Semantics-based	93.0
Pascanu et al. [55]	2015	Echo state networks (ESNs) and recurrent neural networks (RNNs) for feature extraction	Logistic regression and multilayer perceptron classifier	98.3
Garcia [56]	2016	Texture features	Random Forest	95.0
Moshiri et al. [57]	2017	Filter-based feature	Machine learning techniques	99.0
Liu et al. [58]	2017	N-gram-based texture feature	Shared nearest neighbor (SNN) clustering algorithm	98.9
Cakir et al. [59]	2018	Deep learning-based feature	Gradient boosting	96.0
Kalash et al. [60]	2018	GIST features	CNN-based architecture	98.5
Naeem et al. [61]	2019	Local and global malware pattern (LGMP) features	SVM, KNN	98.0
Cui et al. [12]	2019	CNN-based features	CNN	97.6
Naeem et al. [62]	2019	Combined local and global malware (CLGM) features	DCNN	98.18
This paper	**2020**	**Fused SFTA and deep network features**	**DCNN (AlexNet, Inception v3)**	**99.3**

Complexity: As shown in Figure 1, our proposed method consists of five key steps such as data augmentation, features extraction, fusion, selection, and classification. From these, the fusion and selection steps are much important, and the efficiency of our system depends on these. We compute the complexity in the form of Big-O (O) notation for each step. For data augmentation, features extraction, and classification, the complexity is $O(1)$, while for features fusion and selection, the complexity is $O(n) + 3$ and $O(n) + k$, where k denotes the number of selected parameters. Hence, our method is working under one major loop, so the overall complexity is $O(n) = C$.

5. Conclusions

In this paper, we have demonstrated a hybrid method which uses image augmentation and a fusion of segmentation-based fractal texture analysis (SFTA) and pre-trained deep neural network (AlexNet and Inception-v3) features to classify malware images into various classes using different state-of-the-art classifiers. In the first step, the grayscale images are generated using the bytecodes from the malicious programs. Then, in the second step, image augmentation is performed to balance the malware classes and resize the images to $227 \times 227 \times 3$. In the last phase, the pre-trained network features and SFTA best features are selected and fused. Then, the selected features are forwarded to several versions of KNN, SVM, DT, and LD classifiers for malware image identification. The key advantage of the proposed method is the ability to achieve a high level of classification accuracy when compared to the existing methods. When faced with imbalanced data sets there is no one stopping solution to improve the accuracy of the prediction model. In order to get rid of the imbalanced data sets problem, we perform augmentation and balance the dataset and improve the performance of the proposed model. Malware classification based on deep learning introduced an extraordinary improvement when compared with the existing strategies and the experiment result proves the efficiency of the proposed technique as the accuracy was improved and model loss was decreased.

The proposed method achieved an accuracy of 99.3% on the cubic SVM classifier, which displays exceptional performance when compared to other current malware classification approaches.

Author Contributions: Conceptualization, M.R. and M.A.K.; methodology, M.R. and M.A.K.; software, M.N., J.H.S., S.K. and M.A.K.; validation, M.N., J.H.S., S.K., M.R., M.A.K., and R.D.; formal analysis, M.A.K., R.D. and T.B.; investigation, M.N., J.H.S., S.K., M.R., M.A.K., and R.D.; resources, M.N., J.H.S., S.K., M.R., and M.A.K.; data curation, M.N., J.H.S., and S.K.; writing—original draft preparation, M.R. and M.A.K.; writing—review and editing, R.D. and T.B.; visualization, M.N., J.H.S., S.K., M.R., M.R., M.A.K., and R.D.; supervision, M.A.K.; funding acquisition, T.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Symantec. Internet Security Threat Report (ISTR), Technical Report. 2019. Available online: https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf (accessed on 1 June 2020).
2. Alsoghyer, S.; Almomani, I. Ransomware Detection System for Android Applications. *Electronics* **2019**, *8*, 868. [CrossRef]
3. Chun, S.-H. E-Commerce Liability and Security Breaches in Mobile Payment for e-Business Sustainability. *Sustainability* **2019**, *11*, 715. [CrossRef]
4. Wangen, G. The Role of Malware in Reported Cyber Espionage: A Review of the Impact and Mechanism. *Information* **2015**, *6*, 183–211. [CrossRef]
5. Subairu, S.O.; Alhassan, J.; Misra, S.; Abayomi-Alli, O.; Ahuja, R.; Damasevicius, R.; Maskeliunas, R. An Experimental Approach to Unravel Effects of Malware on System Network Interface. In *Lecture Notes in Electrical Engineering*; Springer: Singapore, 2019; pp. 225–235. [CrossRef]
6. Odusami, M.; Abayomi-Alli, O.; Misra, S.; Shobayo, O.; Damasevicius, R.; Maskeliunas, R. Android Malware Detection: A Survey. In *International Conference on Applied Informatics, ICAI, Proceedings of the Communications in Computer and Information Science, Bogotá, Colombia, 1–3 November 2018*; Springer International Publishing: New York, NY, USA, 2018; Volume 942, pp. 255–266. [CrossRef]
7. Vinayakumar, R.; Alazab, M.; Soman, K.P.; Poornachandran, P.; Venkatraman, S. Robust Intelligent Malware Detection Using Deep Learning. *IEEE Access* **2019**, *7*, 46717–46738. [CrossRef]
8. Aslan, O.; Samet, R. A Comprehensive Review on Malware Detection Approaches. *IEEE Access* **2020**, *8*, 6249–6271. [CrossRef]
9. Pan, S.J.; Yang, Q. A Survey on Transfer Learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
10. Kancherla, K.S.; Mukkamala, S. Image visualization based malware detection. In Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Singapore, 16–19 April 2013; pp. 40–44. [CrossRef]
11. Vasan, D.; Alazab, M.; Wassan, S.; Naeem, H.; Safaei, B.; Zheng, Q. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Comput. Netw.* **2020**, *171*, 107138. [CrossRef]
12. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.; Chen, J. Detection of Malicious Code Variants Based on Deep Learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]
13. Ye, Y.; Li, T.; Adjeroh, D.; Iyengar, S.S. A Survey on Malware Detection Using Data Mining Techniques. *ACM Comput. Surv.* **2017**, *50*, 1–40. [CrossRef]
14. Kaur, H.; Pannu, H.S.; Malhi, A.K. A Systematic Review on Imbalanced Data Challenges in Machine Learning. *ACM Comput. Surv.* **2019**, *52*, 1–36. [CrossRef]
15. Costa, A.F.; Humpire-Mamani, G.; Traina, A.J.M. An Efficient Algorithm for Fractal Analysis of Textures. In Proceedings of the 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images, Ouro Preto, Brazil, 22–25 August 2012; pp. 39–46. [CrossRef]
16. Khan, M.A.; Javed, K.; Khan, S.A.; Saba, T.; Habib, U.; Khan, J.A.; Abbasi, A.A. Human action recognition using fusion of multiview and deep features: An application to video surveillance. *Multimed. Tools Appl.* **2020**, 1–27. [CrossRef]

17. Arshad, H.; Khan, M.A.; Sharif, M.I.; Yasmin, M.; Tavares, J.M.R.S.; Zhang, Y.D.; Satapathy, S.C. A multilevel paradigm for deep convolutional neural network features selection with an application to human gait recognition. *Expert Syst.* **2020**, e12541. [CrossRef]
18. Mehmood, A.; Khan, M.A.; Sharif, M.; Khan, S.A.; Shaheen, M.; Saba, T.; Riaz, N.; Ashraf, I. Prosperous Human Gait Recognition: An end-to-end system based on pre-trained CNN features selection. *Multimed. Tools Appl.* **2020**. [CrossRef]
19. Rashid, M.; Khan, M.A.; Alhaisoni, M.; Wang, S.H.; Naqvi, S.R.; Rehman, A.; Saba, T. A Sustainable Deep Learning Framework for Object Recognition Using Multi-Layers Deep Features Fusion and Selection. *Sustainability* **2020**, *12*, 5037. [CrossRef]
20. Hussain, N.; Khan, M.A.; Sharif, M.; Khan, S.A.; Albesher, A.A.; Saba, T.; Armaghan, A. A deep neural network and classical features based scheme for objects recognition: An application for machine inspection. *Multimed Tools Appl.* **2020**. [CrossRef]
21. Rauf, H.T.; Shoaib, U.; Lali, M.I.; Alhaisoni, M.; Irfan, M.N.; Khan, M.A. Particle Swarm Optimization WITH Probability Sequence for Global Optimization. *IEEE Access* **2020**, *8*, 110535–110549. [CrossRef]
22. Khan, M.A.; Khan, M.A.; Ahmed, F.; Mittal, M.; Goyal, L.M.; Jude Hemanth, D.; Satapathy, S.C. Gastrointestinal diseases segmentation and classification based on duo-deep architectures. *Pattern Recognit. Lett.* **2020**, *131*, 193–204. [CrossRef]
23. Sharif, M.I.; Li, J.P.; Khan, M.A.; Saleem, M.A. Active deep neural network features selection for segmentation and recognition of brain tumors using MRI images. *Pattern Recognit. Lett.* **2020**, *129*, 181–189. [CrossRef]
24. Namavar Jahromi, A.; Hashemi, S.; Dehghantanha, A.; Choo, K.-K.R.; Karimipour, H.; Newton, D.E.; Parizi, R.M. An improved two-hidden-layer extreme learning machine for malware hunting. *Comput. Secur.* **2020**, *89*, 101655. [CrossRef]
25. Zhu, D.; Jin, H.; Yang, Y.; Wu, D.; Chen, W. DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data. In Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–7 July 2017; pp. 438–443. [CrossRef]
26. Jeon, S.; Moon, J. Malware-Detection Method with a Convolutional Recurrent Neural Network Using Opcode Sequences. *Inf. Sci.* **2020**, *535*, 1–15. [CrossRef]
27. Sung, Y.; Jang, S.; Jeong, Y.-S.; Park, J.H. Malware classification algorithm using advanced Word2vec-based Bi-LSTM for ground control stations. *Comput. Commun.* **2020**, *153*, 342–348. [CrossRef]
28. Gibert, D.; Mateu, C.; Planes, J. HYDRA: A multimodal deep learning framework for malware classification. *Comput. Secur.* **2020**, *95*, 101873. [CrossRef]
29. Venkatraman, S.; Alazab, M.; Vinayakumar, R. A hybrid deep learning image-based analysis for effective malware detection. *J. Inf. Secur. Appl.* **2019**, *47*, 377–389. [CrossRef]
30. Zhong, W.; Gu, F. A multi-level deep learning system for malware detection. *Expert Syst. Appl.* **2019**, *133*, 151–162. [CrossRef]
31. Ye, Y.; Chen, L.; Hou, S.; Hardy, W.; Li, X. DeepAM: A heterogeneous deep learning framework for intelligent malware detection. *Knowl. Inf. Syst.* **2017**. [CrossRef]
32. Yuxin, D.; Siyi, Z. Malware detection based on deep learning algorithm. *Neural Comput. Appl.* **2017**, *31*, 461–472. [CrossRef]
33. Vasan, D.; Alazab, M.; Wassan, S.; Safaei, B.; Zheng, Q. Image-Based malware classification using ensemble of CNN architectures (IMCEC). *Comput. Secur.* **2020**, *92*, 101748. [CrossRef]
34. Čeponis, D.; Goranin, N. Investigation of Dual-Flow Deep Learning Models LSTM-FCN and GRU-FCN Efficiency against Single-Flow CNN Models for the Host-Based Intrusion and Malware Detection Task on Univariate Times Series Data. *Appl. Sci.* **2020**, *10*, 2373. [CrossRef]
35. Billah, E.; Debbabi, M.; Derhab, A.; Mouheb, D. MalDozer: Automatic framework for android malware detection using deep learning. *Digit. Investig.* **2018**, *24*, S48–S59. [CrossRef]
36. Pektaş, A.; Acarman, T. Deep learning for effective Android malware detection using API call graph embeddings. *Soft Comput.* **2019**, *24*, 1027–1043. [CrossRef]
37. D'Angelo, G.; Ficco, M.; Palmieri, F. Malware detection in mobile environments based on Autoencoders and API-images. *J. Parallel Distrib. Comput.* **2020**, *137*, 26–33. [CrossRef]
38. Naeem, H.; Ullah, F.; Naeem, M.R.; Khalid, S.; Vasan, D.; Jabbar, S.; Saeed, S. Malware detection in industrial internet of things based on hybrid image visualization and deep learning model. *Ad Hoc Netw.* **2020**, *105*, 102154. [CrossRef]

39. Vidal, J.M.; Monge, M.A.S.; Villalba, L.J.G. A novel pattern recognition system for detecting Android malware by analyzing suspicious boot sequences. *Knowl. Based Syst.* **2018**, *150*, 198–217. [CrossRef]
40. Kabakus, A.T. What Static Analysis Can Utmost Offer for Android Malware Detection. *Inf. Technol. Control* **2019**, *48*, 235–240. [CrossRef]
41. Narayanan, A.; Chandramohan, M.; Chen, L.; Liu, Y. A multi-view context-aware approach to Android malware detection and malicious code localization. *Empir. Softw. Eng.* **2017**, *23*, 1222–1274. [CrossRef]
42. Du, D.; Sun, Y.; Ma, Y.; Xiao, F. A Novel Approach to Detect Malware Variants Based on Classified Behaviors. *IEEE Access* **2019**, *7*, 81770–81782. [CrossRef]
43. Alam, S.; Qu, Z.; Riley, R.; Chen, Y.; Rastogi, V. DroidNative: Automating and optimizing detection of Android native code malware variants. *Comput. Secur.* **2017**, *65*, 230–246. [CrossRef]
44. Kang, H.; Jang, J.; Mohaisen, A.; Kim, H.K. Detecting and Classifying Android Malware Using Static Analysis along with Creator Information. *Int. J. Distrib. Sens. Netw.* **2015**, *11*, 479174. [CrossRef]
45. Wen, L.; Yu, H. An Android malware detection system based on machine learning. In Proceedings of the 2017 International Conference on Green Energy and Sustainable Development (GESD 2017), Chongqing, China, 27–28 May 2017. [CrossRef]
46. Johnson, J.M.; Khoshgoftaar, T.M. Survey on deep learning with class imbalance. *J. Big Data* **2019**, *6*. [CrossRef]
47. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS'12, Lake Tahoe, NV, USA, 3–6 December 2012; Curran Associates Inc.: Red Hook, NY, USA, 2012; Volume 1, pp. 1097–1105.
48. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
49. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
50. Mikolajczyk, A.; Grochowski, M. Data augmentation for improving deep learning in image classification problem. In Proceedings of the International Interdisciplinary PhD Workshop (IIPhDW), Świnoujście, Poland, 9–12 May 2018. [CrossRef]
51. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec '11, Art. No. 4, Pittsburgh, PA, USA, 20 July 2011. [CrossRef]
52. Anderson, B.; Storlie, C.; Lane, T. Improving malware classification. In Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence-AISec, Raleigh, NC, USA, 12 October 2012. [CrossRef]
53. Dahl, G.E.; Stokes, J.W.; Deng, L.; Yu, D. Large-scale malware classification using random projections and neural networks. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, 26–31 May 2013. [CrossRef]
54. Zhang, M.; Duan, Y.; Yin, H.; Zhao, Z. Semantics-Aware Android Malware Classification Using Weighted Contextual API Dependency Graphs. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, Scottsdale, AZ, USA, 3–7 November 2014. [CrossRef]
55. Pascanu, R.; Stokes, J.W.; Sanossian, H.; Marinescu, M.; Thomas, A. Malware classification with recurrent networks. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, QLD, Australia, 17–24 April 2015. [CrossRef]
56. Garcia, F.C.C. Random Forest for Malware Classification. *arXiv* **2016**, arXiv:1609.07770. Available online: https://arxiv.org/abs/1609.07770 (accessed on 1 June 2020).
57. Moshiri, E.; Abdullah, A.B.; Azlina, R.; Raja, B.; Muda, Z. Malware Classification Framework for Dynamic Analysis using Information Theory. *Indian J. Sci. Technol.* **2017**, *10*, 1–10. [CrossRef]
58. Liu, L.; Wang, B.; Yu, B.; Zhong, Q. Automatic malware classification and new malware detection using machine learning. *Front. Inf. Technol. Electron Eng.* **2017**, *18*, 1336–1347. [CrossRef]
59. Cakir, B.; Dogdu, E. Malware classification using deep learning methods. In Proceedings of the ACM Southeast Conference, ACMSE '18, Richmond, VA, USA, 29–31 March 2018; Association for Computing Machinery: New York, NY, USA, 2019; pp. 1–5. [CrossRef]

60. Kalash, M.; Rochan, M.; Mohammed, N.; Bruce, N.D.B.; Wang, Y.; Iqbal, F. Malware Classification with Deep Convolutional Neural Networks. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018. [CrossRef]
61. Naeem, H.; Guo, B.; Naeem, M.R.; Ullah, F.; Aldabbas, H.; Javed, M.S. Identification of malicious code variants based on image visualization. *Comput. Electr. Eng.* **2019**, *76*, 225–237. [CrossRef]
62. Naeem, H. Detection of Malicious Activities in Internet of Things Environment Based on Binary Visualization and Machine. *Wirel. Pers. Commun.* **2019**. [CrossRef]