



Article

Robust Malware Family Classification Using Effective Features and Classifiers

Baraa Tareq Hammad ¹, Norziana Jamil ² , Ismail Taha Ahmed ¹, Zuhaira Muhammad Zain ³
and Shakila Basheer ^{3,*} 

¹ College of Computer Sciences and Information Technology, University of Anbar, Anbar 55431, Iraq

² College of Computing and Informatics, University Tenaga Nasional, Kajang 43000, Selangor, Malaysia

³ Department of Information Systems, College of Computer and Information Science, Princess Nourah Bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia

* Correspondence: sbbasheer@pnu.edu.sa

Abstract: Malware development has significantly increased recently, posing a serious security risk to both consumers and businesses. Malware developers continually find new ways to circumvent security research's ongoing efforts to guard against malware attacks. Malware Classification (MC) entails labeling a class of malware to a specific sample, while malware detection merely entails finding malware without identifying which kind of malware it is. There are two main reasons why the most popular MC techniques have a low classification rate. First, Finding and developing accurate features requires highly specialized domain expertise. Second, a data imbalance that makes it challenging to classify and correctly identify malware. Furthermore, the proposed malware classification (MC) method consists of the following five steps: (i) Dataset preparation: 2D malware images are created from the malware binary files; (ii) Visualized Malware Pre-processing: the visual malware images need to be scaled to fit the CNN model's input size; (iii) Feature extraction: both hand-engineering (Tamura) and deep learning (GoogLeNet) techniques are used to extract the features in this step; (iv) Classification: to perform malware classification, we employed k-Nearest Neighbor (KNN), Support Vector Machines (SVM), and Extreme Learning Machine (ELM). The proposed method is tested on a standard Malimg unbalanced dataset. The accuracy rate of the proposed method was extremely high, making it the most efficient option available. The proposed method's accuracy rate was outperformed both the Hand-crafted feature and Deep Feature techniques, at 95.42 and 96.84 percent.

Keywords: malware classification; deep learning; Tamura; GoogLeNet; KNN; SVM; ELM



Citation: Hammad, B.T.; Jamil, N.; Ahmed, I.T.; Zain, Z.M.; Basheer, S. Robust Malware Family Classification Using Effective Features and Classifiers. *Appl. Sci.* **2022**, *12*, 7877. <https://doi.org/10.3390/app12157877>

Academic Editor: Luis Javier Garcia Villalba

Received: 1 July 2022

Accepted: 21 July 2022

Published: 5 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Security is the body of procedures, methods, techniques, and technologies that come together to defend the availability, confidentiality, and integrity of information from intrusion. The information is protected by a variety of defense mechanisms, including firewalls, antivirus software systems, and others. Nevertheless, many attackers continue to attempt to log into any system by identifying a single vulnerability in the system that require protection [1].

Software programs known as malware (malicious software) are intended to harm or carry out any kind of undesired activity on a computer system, such as interfering with computer operations, gathering confidential details, evading security controls, and displaying undesirable ads. Massive amounts of malware are purposefully produced every day. The market for malicious software has grown to be more expensive and is dynamically expanding yearly. Based on how it operates, Figure 1 illustrates the various classes of malware, including (adware, spyware, virus, trojan, worm, and backdoor, among others) [2,3].

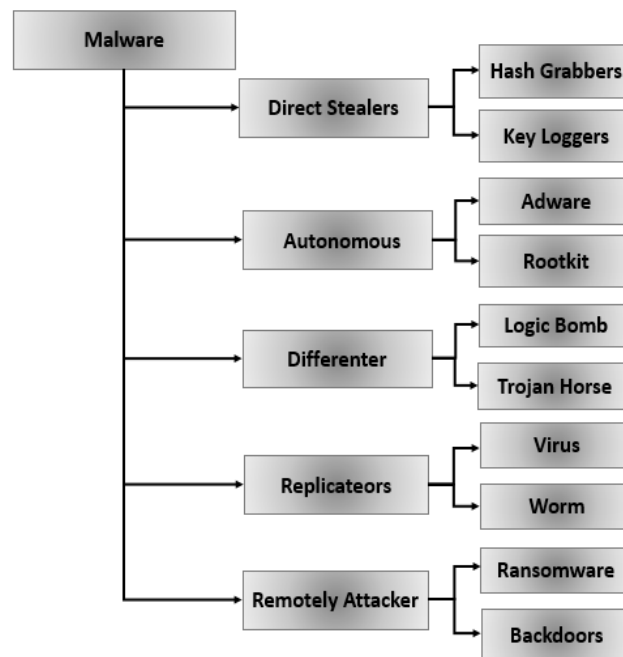


Figure 1. Malware Classes [3].

The previous methods are unable to guard against the rise in malware attacks, which led numerous researchers to suggest various ways for classifying malware. Figure 2 shows a general taxonomy of malware classification (MC) across several domain types.

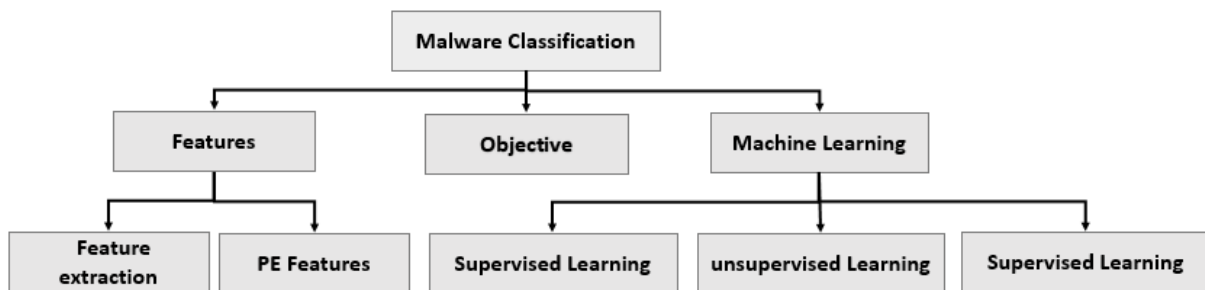


Figure 2. Malware Classification Taxonomy.

As illustrated in Figure 3, most current MC approaches can be divided into two categories: hand-crafted and deep learning-crafted.

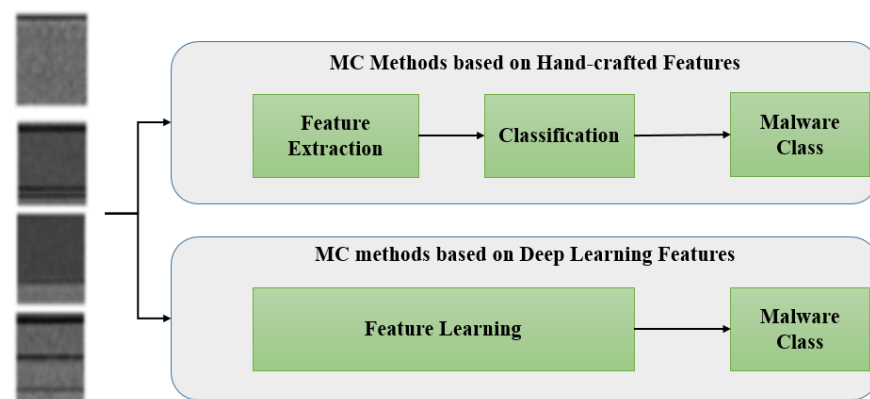


Figure 3. Malware classification techniques across a variety of features kinds.

Hand-crafted features are a common feature of conventional MC techniques. The manual selection of the feature type is part of the feature extraction process. While the deep learning methods can identify and classify malware automatically.

With the increase in malware, earlier works based on conventional techniques are time-consuming and ineffective. The visualization technique is efficient in terms of computing speed and time. The raw pixel data from photos cannot be handled by traditional machine learning techniques; therefore, increased learning is not possible. Raw data must be converted into feature vectors, which requires a high level of engineering and technical expertise. The converted form of the input photos is trained by the classification model. Deep learning approaches allow automatic learning by enabling representation learning to employ unprocessed input data [4].

Convolutional Neural Networks (CNNs), which are employed in image classification, are among the most well-known deep learning techniques. Because CNN offers a better representation of the data, manual feature design can be ignored. The image data is first transformed into a pixel array. The pixel array is then analyzed by a number of convolutional layers, which ultimately produce an expected output. Largest and most well-known datasets, including ImageNet, are used to train the CNN models. However, the information learned from CNNs can be transferred to various learning tasks to improve malware classification and detection. The main benefit of transfer learning would be that it permits task training with a small dataset by employing a model that has already been trained on a big dataset.

Most of the current MC methods generally have numerous serious problems: (1) Finding accurate and convenient features for malware classification, (2) the creation of the features requires extensive feature engineering or highly specialized domain expertise, and (3) a poor rate of classification accuracy, and (4) an unbalance in the data that makes it difficult to classify and accurately identify malware. Therefore, the proposed malware classification method consists of the following five steps: (i) dataset preparation: 2D malware images are created from the malware binary files, (ii) visualized malware pre-processing: the visual malware images need to be scaled to fit the CNN model's input size, (iii) feature extraction: both hand-engineering (Tamura) and deep learning (GoogLeNet) techniques are used to extract the features in this step, and (iv) classification: to perform malware classification, we employed KNN, SVM, and ELM.

The following are the contributions of the proposed method:

1. Using the Tamara and deep methods, feature vectors are extracted in order to propose an effective malware classification method.
2. It is proposed to use a malware visualization technique that produces grayscale graphics by converting binary files into 8-bit vectors.
3. Classify malware with a visual component that employs a pre-trained CNN model that can identify visual malware samples without the need for features engineering.
4. Using optimized CNN models to operate correctly on 9339 photos of 25 different malware families comprising eight distinct malware kinds.
5. Seeking to establish a CNN model that can correctly classify malware using unbalanced dataset (e.g., Malimg dataset).

The rest of this article is structured as follows. Section 2 discusses the relevant works. Section 3 provides a description of the various features. Section 4 describes the proposed techniques. Section 5 discusses findings and analyses. Finally, conclusions can be taken from Section 6.

2. Related Works

Based on various analysis methods, numerous malware classification (MC) and identification methods have been developed. These works fall into two categories: Methods based on hand-crafted features and methods based on deep learning features. A thorough analysis of several malware classification techniques is provided in this section.

Nataraj et al. [4] proposed an MC method based on GIST features from the visualization grayscale images. The proposed algorithm experimented on Maling dataset. In the classification scenario, the K-nearest neighbor is the classifier that is applied. Barath et al. [5] proposed an MC method based on global features using the principal component analysis (PCA). In the classification scenario, the nearest neighbor is the classifier that is applied. Kosmidis and Kalloniatis [6] proposed an MC method based on the GIST feature extraction technique. In the classification scenario, random forest is the classifier applied. Naeem et al. [7] proposed an MC method based on the D-SIFT and GIST feature extraction methods. In the classification scenario, Nearest Neighbor (KNN) and Support Vector Machine (SVM) are the two classifiers applied. Makandar et al. [8] proposed an MC method based on Gabor Wavelet, GIST, and Discrete wavelet Transform. The proposed algorithm experimented on Maling dataset. In the classification scenario, SVM is the classifier applied. Verma et al. [9] proposed an MC method based on a combination of the first-order and grey-level co-occurrence matrix (GLCM)-based second-order statistical texture features. The proposed algorithm experimented on Maling dataset. In the classification scenario, ELM is the classifier applied.

Deep learning approaches are currently being applied to malware classification issues to get beyond the challenging feature engineering effort or the need for domain expertise. CNN consider one of the popular methods of DL. CNN architectures have been used in several research studies on the classification of malware. Some authors created their unique CNN models without using any pre-trained models. Sun and Qian [10] proposed an MC method based on recurrent neural network (RNN) and CNN models. The proposed algorithm experimented on the Maling dataset. In the classification scenario, CNN is the classifier applied. Gilbert et al. [11] proposed an MC method based on CNN with three convolutional layers with one fully connected layer. The proposed algorithm experimented on the Maling dataset. Agarap et al. [12] proposed an MC method based on the DenseNet model. The proposed algorithm experimented on the Maling dataset. In the classification scenario, SVM is the classifier applied. Çayır et al. [13] proposed an MC method based on SIMPLE capsule networks (CapsNet). Additionally, CapsNet does not require transfer learning, thus, starting from scratch with training the model is simple. The Maling dataset was used for testing the proposed approach. Ensemble is the classifier used in the classification scenario. Gibert et al. [14] proposed an MC method based on 2D CNN. The Maling dataset was used for testing the proposed approach. David and Netanyahu [15] proposed an MC method based on DBNs and denoising autoencoders. In the classification scenario, SVM is the classifier applied.

To enhance the classification procedure, some researchers may decide to integrate multiple training models. Nisa et al. [16] proposed a malware classification (MC) method based on combining two models of deep CNN (DCNN), AlexNet, and Inception-v3, with SFTA (scale feature texture analyzer). In the classification scenario, machine learning classifiers were used. Vasan et al. [17] proposed an MC method based on combining two models of deep CNN, VGG16, and ResNet50 models. In the classification scenario, Softmax and Multiclass SVMs are the two classifiers applied. The dimensionality of the features was then reduced using a PCA (principal component analysis) approach. El-Shafai et al. [18] proposed an MC method based on pretrained fine-tuned CNN models (AlexNet, DenseNet-201, DarkNet-53, ResNet-50, Inception-V3, VGG16, MobileNet-V2, and Places365-GoogleNet.). The proposed algorithm experimented on Maling dataset. Vasan et al. [19] proposed an MC method based on combining two models of deep CNN, GoogleNet, and ResNet models. Bennasar et al. [20] proposed an MC method based on combining three models of deep CNN, AlexNet, VGG, GoogleNet, and ResNet. Jeyaprakash et al. [2] proposed an MC method based on DenseNet model. The proposed algorithm experimented on Maling dataset. In the classification scenario, a fully connected softmax layer is applied.

In other cases, the CNN could be used to extract features when machine learning methods are being used to classify malware. Roseline et al. [21] proposed an MC method based on deploying a sequential multilayered Random Forest ensemble technique. Fol-

lowing that, four distinct machine learning (ML) approaches were used: Random forests (RF), Xgboost, Extra trees classifier (ETC), and Logistic regression (LR). Ouahab et al. [22] proposed an MC method based similarities among the malware images. In the classification scenario, k-Nearest Neighbor (KNN) is the classifier that have been applied. Agarap and Pepito [12] proposed an MC method based on CNN features. CNN-SVM, GRU-SVM, and MLP-SVM models are used to classify each malware family. Awan et al. [23] proposed an MC method based on spatial attention and convolutional neural network (SACNN) and deep learning frameworks (VGG19 model). The proposed algorithm experimented on the Maling dataset. In the classification scenario, a fully connected softmax layer was applied. Sudhakar and Kumar [24] proposed an MC method based on an updated ResNet50 model. The proposed algorithm experimented on the Maling dataset. In the classification scenario, a fully connected softmax layer was applied. Xiao et al. [25] proposed an MC method based on structural entropy and deep CNNs. The proposed algorithm experimented on the Maling dataset. In the classification scenario, SVM is the classifier applied. Most previously mentioned MC techniques have a low classification rate attributable to the following factors. First, it takes a highly specialized level of domain knowledge to find and produce accurate features. Second, makes it challenging to classify and correctly identify malware. Therefore, the proposed malware classification method will be employed to address the aforementioned issues.

3. Multiple Features

Color, texture, and shape are examples of features that can be extracted. The main function of feature extraction is to collect the visible info of an image and save it as feature vectors. There are different kinds of feature design strategies: conventional methods typically depend on hand-crafted features and deep learning techniques (DL). Hand-crafted features are done manually by a human. While DL automatically extracts the features.

3.1. Tamura Feature

One of the most prevalent merits of different image types' analysis is texture. Texture has properties like regularity and scale, and it can be characterized by different orientations, coarseness, contrast, and other factors [18]. Texture plays a big role in human eyesight. Even though everyone can identify texture, defining it might be difficult. As a result, image texture-based classification [4] has recently been utilized to classify malware.

Tamura is one of the most popular texture features. Tamura and colleagues devised texture features that correlate to human visual perception [18]. They evaluated psychological assessments for human volunteers to six textural features (coarseness, contrast, directionality, line-likeness, regularity, and roughness). The first three achieved excellent results and included coarseness, contrast, and directionality.

An image will have textures at various scales; coarseness seeks to find the biggest size from which a texture occurs, even when a tiny micro-texture is present. In terms of computation, one calculates averages at each point across neighborhoods with linear sizes that are powers of two. At the point (x, y) , the mean over the neighborhood of size $2^k \times 2^k$ equals:

$$A_k(x, y) = \frac{\sum_{i=x-2^{k-1}}^{x+2^{k-1}-1} \sum_{j=y-2^{k-1}}^{y+2^{k-1}-1} f(i, j)}{2^{2k}} \quad (1)$$

where, $2^k \times 2^k$ size is the average of neighborhood.

$$E_{k,h}(x, y) = \left| A_k(x + 2^{k-1}, y) - A_k(x - 2^{k-1}, y) \right| \quad (2)$$

The difference between these two means that they belong to non-overlapping neighborhoods is calculated using Equation (2).

Contrast is a metric that determines how the distribution of gray levels fluctuates in an image and how skewed it is toward black or white. The standard deviation of gray levels is

first calculated. The kurtosis α_4 is then used in the second step. As a result, the contrast measure is defined as:

$$\text{Contrast} = \sigma / (\alpha_4) \quad (3)$$

$$\alpha_4 = \mu_4 / \sigma^4 \quad (4)$$

when the variance is α_2 and the fourth moment around the mean is μ_4 . The closest value, according per Tamura, is $n = \frac{1}{4}$.

Directionality is a texture characteristic that affects the entire image. The presented feature does not attempt to distinguish between various orientations or patterns but rather the overall degree of directionality. A scale of 0 to 1 was employed to determine the degree of directionality. As a result, two patterns with the same degree of directionality but different orientations are deemed the same. Equation (5) was used to calculate the directionality.

$$\text{Directionality} = 1 - rn_{peaks} \sum_{p=1}^{n_{peaks}} \sum_{a \in w_p} (a - a_p)^2 H \text{directionality}(a) \quad (5)$$

where r signifies a normalizing factor associated to quantizing levels of the angle a , n indicates quantization direction angle, np represents the number of peaks, ap indicates the location of the peaks, w_p indicates the range of the angles assigned to the Pth peak, and the histogram of quantization direction value known as $H\text{Directionality}$ is created by calculating the number of edge pixels that have associated directional angles.

3.2. GoogleNet Deep Feature

Due to its multilayered design, Deep Learning can learn the features of both labeled and unlabeled input. However, the model must be trained on a substantial amount of data each time, which consumes time and computer power. To get around this learning challenge, we can employ pre-trained deep learning designs to gather features for the classification model. This inspires us to create pre-trained network-based deep learning for identifying and classifying different types of malware.

There are Numerous Pre-trained CNN models such as DenseNet-201, GoogleNet, InceptionV3, Xception, Res-net-18, ResNet-50, AlexNet, VGG16 and VGG19 [26,27]. These Pre-trained CNN models have been widely applied to a variety of image multiclass classification, including the classification of malware families by numerous investigators [28]. Among the already-built customized CNNs, GoogleNet [29] has been selected as the best.

GoogleNet is a well-known convolutional neural network (CNN) constructed using the Inception architecture. There are 22 layers in the GoogleNet design (27 layers, including pooling layers). A 224×224 image with RGB color channels is fed into the GoogLeNet as an input layer. The purpose of the Inception modules is to provide the network with a range of convolutional filter sizes in each block. The main layout of GoogleNet is depicted in Figure 4. Rectified Linear Units (ReLU) serve as the activation functions for each convolution in this structure. Compared to earlier state-of-the-art architectures like AlexNet and ZF-Net, Google's architecture is quite distinctive. They used a 22-layer deep CNN as part of their design. However, because the module relied on a number of tiny convolutions, they had to cut the number of parameters from 60 million (AlexNet) to 4 million. The network was created with practicality and computational efficiency in mind, allowing inference to be executed on individual devices, even ones with constrained computational capabilities, particularly those with small memory footprints.

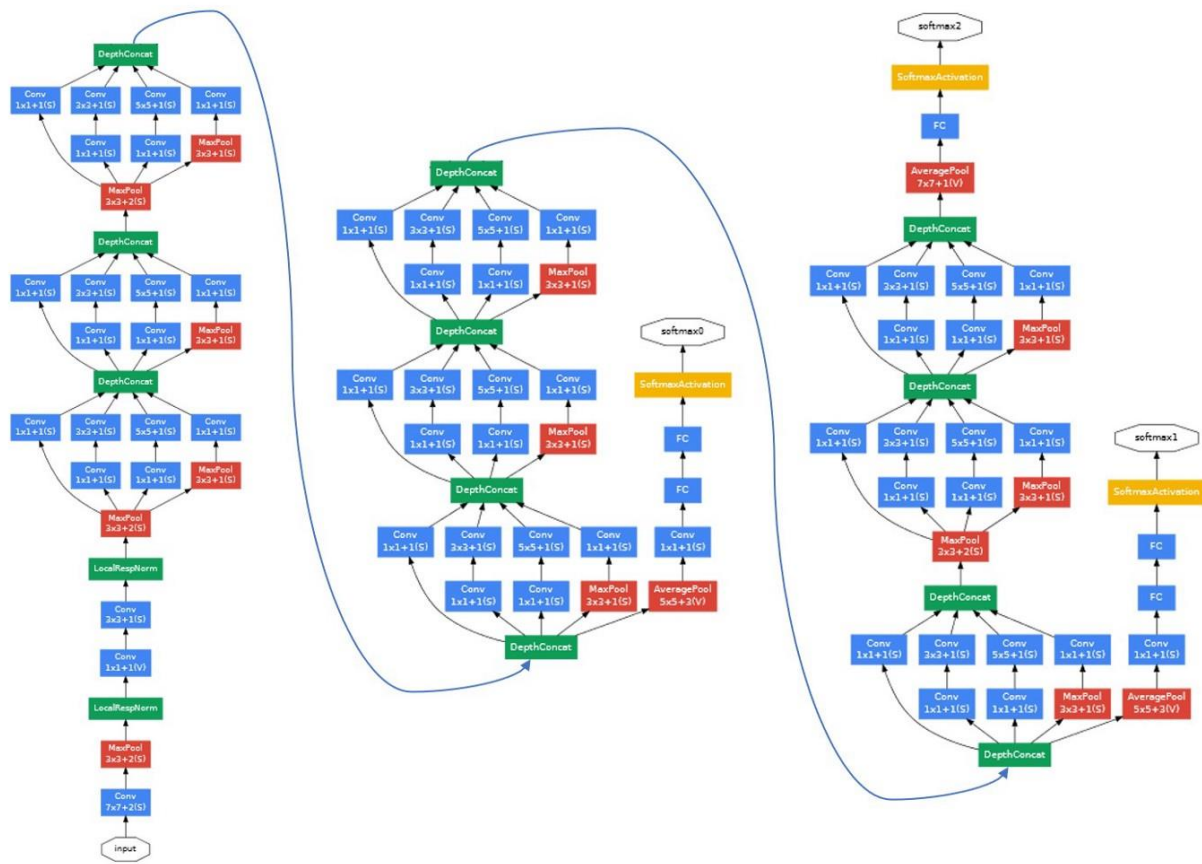


Figure 4. GoogLeNet design [29].

4. The Proposed Method

Figure 5 depicts the fundamental steps of the proposed malware classification method. There are five steps in total: (1) Dataset preparation, (2) Visualized Malware Pre-processing, (3) Feature extraction, and (4) Classification. These steps' specifics are as follows:

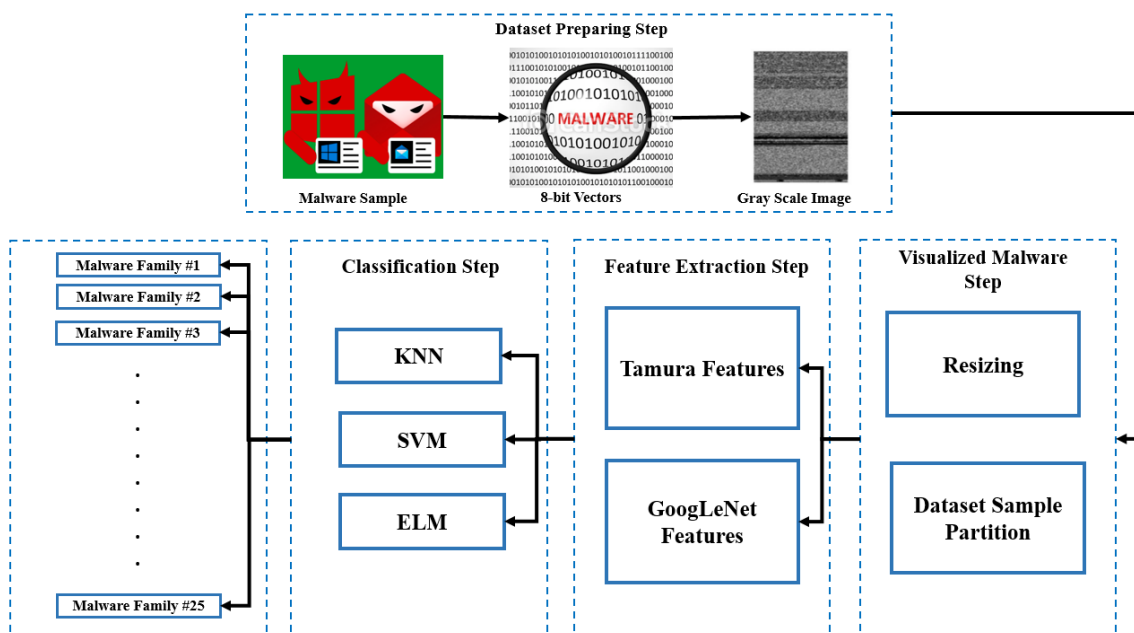


Figure 5. Proposed method flowchart.

Step 1: Dataset Preparing.

Malware binaries are used to construct eight-bit vectors. A grayscale image, as seen in Figure 6, is created from the resultant 8-bit vectors. This procedure is known as visualization. The ability to clearly distinguish between the various parts of a binary is the greatest advantage of malware visualization. Visualization-based malware classification is both substantially faster and more accurate than conventional analysis techniques. Figure 7 displays the bytes file, which represents the malware’s binary data in its raw hexadecimal format provided by Microsoft. As evident from the figure, the first eight characters of each line are used as a reference to identify the line number. The final line demonstrates how certain hex content is inaccessible and is represented by “??”.

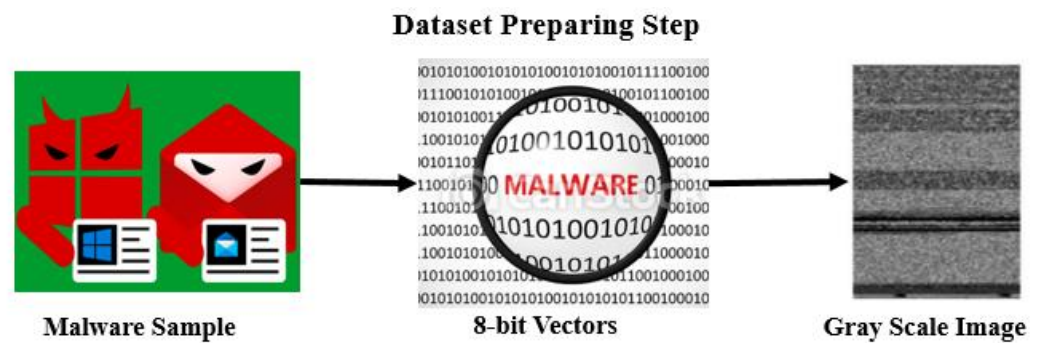


Figure 6. Malware binary to grayscale image conversion procedure.

00401000	56	8D	44	24	08	50	8B	F1	E8	1C	1B	00	00	C7	06	08
00401010	BB	42	00	8B	C6	5E	C2	04	00	CC	CC	CC	CC	CC	CC	CC
00401020	C7	01	08	BB	42	00	E9	26	1C	00	00	CC	CC	CC	CC	CC
00401030	56	8B	F1	C7	06	08	BB	42	00	E8	13	1C	00	00	F6	44
00401040	24	08	01	74	09	56	E8	6C	1E	00	00	83	C4	04	8B	C6
00401050	5E	C2	04	00	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
00401060	8B	44	24	08	8A	08	8B	54	24	04	88	0A	C3	CC	CC	CC
00401070	8B	44	24	04	8D	50	01	8A	08	40	84	C9	75	F9	2B	C2
00401080	C3	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
00401090	8B	44	24	10	8B	4C	24	0C	8B	54	24	08	56	8B	74	24
004010A0	08	50	51	52	56	E8	18	1E	00	00	83	C4	10	8B	C6	5E
004010B0	C3	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC	CC
0042A800	??	??	??	??	??	??	??	??	??	??	??	??	??	??	??	??

Figure 7. One byte file screenshot.

Step 2: Visualized Malware Pre-processing.

The malware binary files are transformed into 2D malware images, as was mentioned in the first phase, and these images’ sizes vary between the 25 families that have been put to the test. The malware images must be resized as part of the pre-processing step for malware information to make the malware images collected from the first step fit with the input size of the used CNN model. The primary benefit of the resizing method is shrinking the size of the input images, which is very helpful in hastening training and lowering the computing burden on the utilized CNN model. Additionally, during the re-dimensionalization procedure, the malware images’ primary textural characteristics are maintained.

Step 3: Feature Extraction.

In the approaches used to classify malware, the feature extraction step is essential. Both hand-engineering and deep learning techniques are used to extract the features in this

case. As described in the next subsections, the objective is to extract valuable features from different applications.

Step 3.1: Tamura Features Extraction.

The malware can be easily identified by its texture-based feature from visualized malware.

Tamura Texture features, including coarseness, contrast, and directionality, are extracted using hand-engineering techniques, as demonstrated in Algorithm 1. The obtained Tamura feature vector has a dimension of 1×3 .

Algorithm 1: Compute Tamura's textures features.

Input: Malware that has been visualized.

Output: Texture features vector.

- 1 Load malware that can be seen.
 - 2 Calculate the texture features to Tamura coarseness, contrast, and directionality according to the Equations (5) and (6).
 - 3 Three features vector are gained.
-

Step 3.2: GoogLeNet Features Extraction.

The deep features were extracted using the GoogLeNet pretrained CNN model, as demonstrated in Algorithm 2. The features were extracted using the activations function on the "pool5-drop 7×7 s1" global average pooling layer. The feature vectors' dimensions are 1×1024 .

Algorithm 2: Compute GoogLeNet features.

Input: Malware that has been visualized.

Output: Deep features vector.

- 1 Load malware that can be seen.
 - 2 The malware images must be resized into 224×224 to fit with the input size of the used GoogLeNet CNN model.
 - 3 Apply GoogLeNet CNN Model.
 - 4 Using the activations function on the global average pooling layer "pool5-drop_{7 × 7_s1}", features were extracted.
 - 5 1×1024 feature vectors are gained.
-

Step 4: Classification.

The remarkable features allow for the creation of an efficient classification model. The primary function of the classification step is to categorize data into classes. Forecasting the category of the supplied data sets is the first step in this procedure. The terms target, label, and classes are frequently used to describe the classes. Numerous studies have rated KNN, Random Forests, ELM, Neural Networks, and Support Vector Machines as having good predicted accuracy results in terms of supervised learning and performances. As practical approaches for malware classification, we used KNN, SVM, and ELM in this paper.

Step 4.1: K-Nearest Neighbor (KNN) Classifier.

Since it is typically a good first option when creating classifiers, the KNN classifier was selected as the first classifier for this research. The easiest and most widely used supervised classifier is the k-Nearest Neighbor (KNN) [30]. It entails categorizing a sample using the k instances from the training set that are closest to it. The Euclidean distance between each feature vector in the training samples and the test sample is determined. Following that, the KNN class labels—where k is an integer—are used to identify the unknown class label. Multiple feature vectors are split into training and testing data sets to create a model, which will be used to create and train the KNN model on the training set. To identify the malware family class for each sample, test the trained KNN model.

Step 4.2: SVM Classifier.

A linear maximum margin classifier called Support Vector Machines (SVM) [31] works by identifying the ideal hyperplane that best distinguishes between two classes. Binary SVM classification can be expanded for multi-class problems by breaking down a multi-class task into numerous binary classification tasks using one-versus-one or one-versus-all approaches. Multiple feature vectors are split into training and testing data sets to create a model, which will be used to create and train the SVM model on the training set. To identify the malware family class for each sample, test the trained SVM model [32].

Step 4.3: Extreme Learning Machines (ELM) Classifier.

An Extreme Learning Machine (ELM) [33] is a simplistic design that connects the input layer connected to the hidden layer and the hidden layer to the output layer. Typically, an ELM contains just one hidden layer. Random assignments are made to the weights and bias matrix between the input and the hidden layer [34]. The aforementioned procedures are computed once. In contrast, weights are regularly adjusted by back-propagation algorithms, which go back and forth. ELMs are incredibly quick for training owing to this one-time computation [34]. Figure 8 shows the ELM design, where the input layer is represented by X , the hidden layer by H , and the output layer by Y .

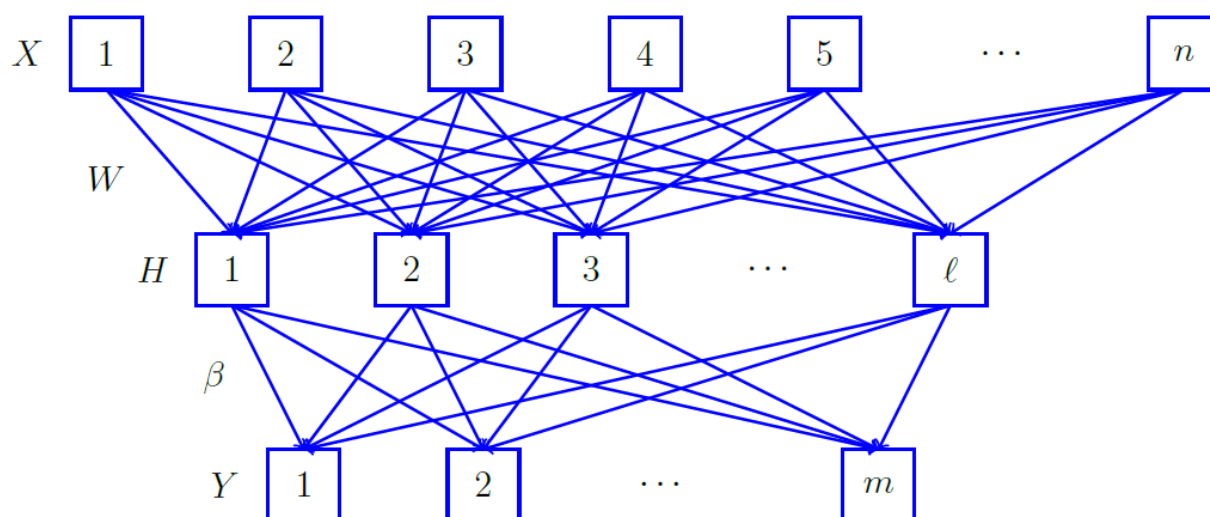


Figure 8. ELM model design [35].

Despite some controversy, experimental investigations have demonstrated that ELMs can deliver respectable predictive performance in some tasks and at a far lower training cost than networks that are trained through back-propagation. The high number of new malware varieties makes ELMs appealing for malware classification. To build a model, multiple feature vectors are divided into two portions (training and testing data sets) that will be utilized to develop and train the ELM model on the training set. Test the trained ELM model to determine the family class of each malware.

5. Results and Discussion

In this section, we test the proposed technique using several indicators and analyze the results. This section includes four subsections, namely datasets, performance evaluation metrics, evaluation results, and comparison with alternative methods. The proposed technique is developed in MATLAB version R2020a on an HP laptop with an Intel Core i7 processor operating at 2.60 GHz and 8 GB of RAM running Microsoft Windows 10 64-bit (OS). The computer ran a 64-bit version of Windows 10.

5.1. Datasets

For academic research, a few malware databases are accessible. Maling [4] is a frequent dataset. The Maling dataset was used to assess the performance of the proposed

technique. The Maling dataset includes 9339 malware photos from 25 different families and classes. According to Table 1, each sample belongs to one of the 25 malware families. Therefore, we aim to create a multi-class categorization of malware. Malware graphics come in a variety of widths and lengths. Various malware families, including Dialer, Backdoor, Worm, Worm-AutoIT, Trojan, Trojan-Downloader, Rouge, and PWS, were used to build the photos. It makes sense that the Maling dataset shares a lot of similarities in terms of texture and size. As a result, classification is simple for a classifier to perform. While variations of some malware families are shown to be extremely unique in other datasets. There is a substantial imbalance in the Maling dataset. More than 30% of the photos fall under class 2 (Allaple.A), whereas 17% fall under class 3 (Allaple.L) as shown in Figure 9. The Maling dataset's malware samples are represented as grayscale pictures with a range of (0: black—255: white). Figure 10 illustrates diverse image textures from distinct malware binary file segments.

Table 1. Description of the Maling dataset families.

Class ID	Family Name	Description	
		Malware Kind	Sample No.
#1	Adialer.C	Dialer	122
#2	Agent.FYI	Backdoor	116
#3	Allaple.A	Worm	2949
#4	Allaple.L	Worm	1591
#5	Alueron.gen!J	Trojan	198
#6	Autorun.K	Worm	106
#7	C2LOP.P	Trojan	200
#8	C2LOP.gen!g	Trojan	146
#9	Dialplatform.B	Dialer	177
#10	Dontovo.A	Downloader	162
#11	Fakerean	rogue	381
#12	Instantaccess	Dialer	431
#13	Lolyda.AA1	PWS	213
#14	Lolyda.AA2	PWS	184
#15	Lolyda.AA3	PWS	123
#16	Lolyda.AT	PWS	159
#17	Malex.gen!J	Trojan	136
#18	Obfuscator.AD	Downloader	142
#19	Rbot!gen	Backdoor	158
#20	Skintrim.N	Trojan	80
#21	Swizzor.gen!E	Downloader	128
#22	Swizzor.gen!I	Downloader	132
#23	VB.AT	Worm	408
#24	Wintrim.BX	Downloader	97
#25	Yuner.A	Worm	800
	Total	-	9339

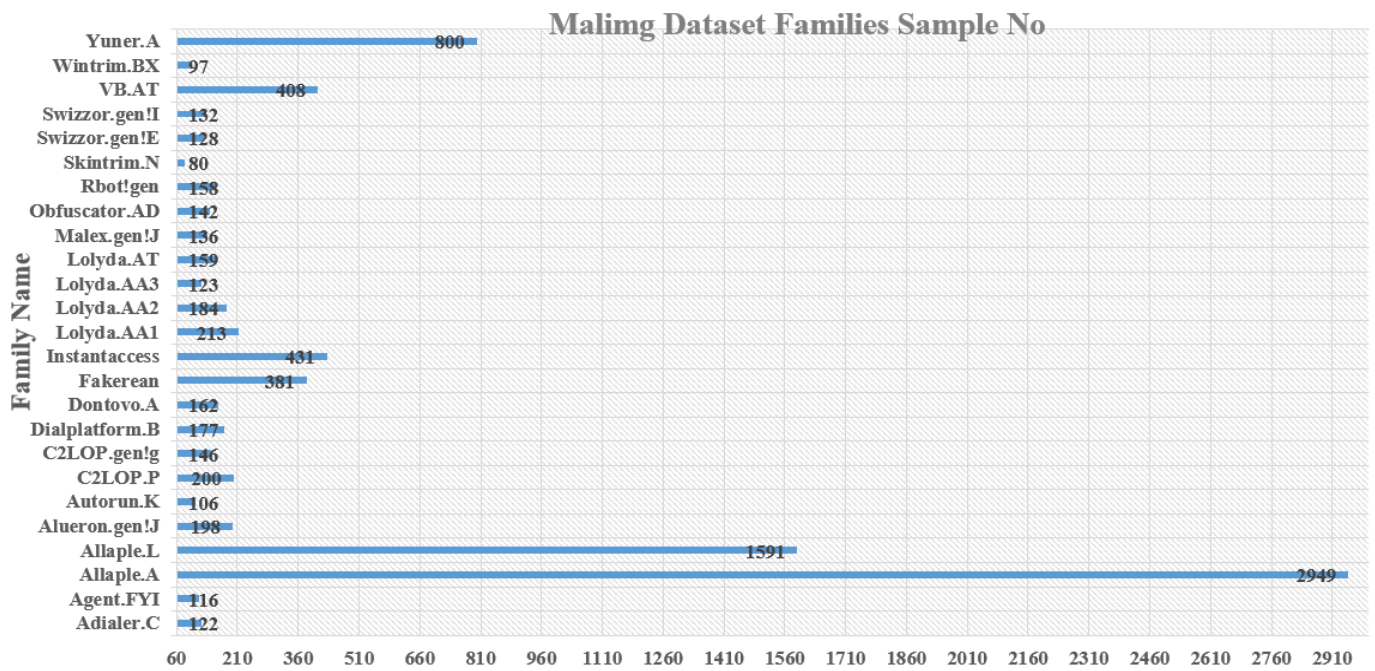


Figure 9. Dispersion of malware across classes inside the Maling dataset.

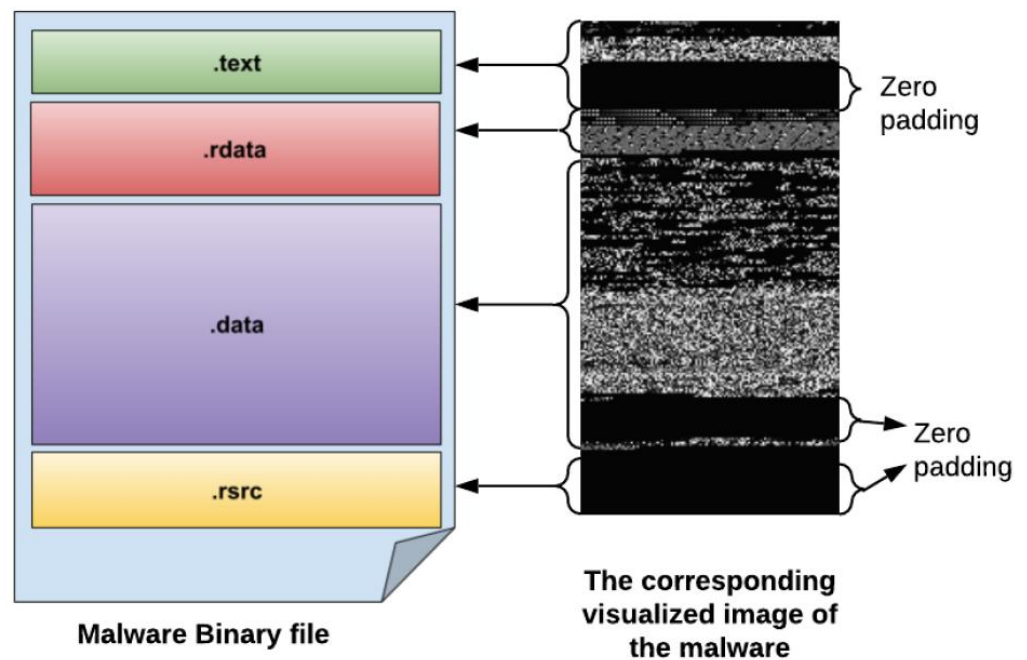


Figure 10. A sample of the visualized version of malware binary file segments and their associated parts.

5.2. Performance Evaluation Metric

The proportion of samples that are successfully identified to all items is known as classification accuracy (CAc). When the classes are uneven, accuracy is less helpful. When the classes are balanced, it offers helpful information [36]. The following formula is used to calculate Classification Accuracy:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \times 100 \% \tag{6}$$

Both existing and expected malware kinds might be viewed as positive in order to calculate the *TP* (true positive). Both existing and expected malware kinds might be viewed as negative in order to calculate the *TN* (true negative). In order to compute the *FP* (false positive), the present malware kinds might be seen as negative, and the expected malware kinds might be seen as positive. In order to determine the *FN* (false negative), the existing malware kinds may be seen as positive and the expected malware kinds as negative. Figure 11 illustrates an estimation of the *TP*, *TN*, *FP*, and *FN* parameters. In contrast to the conventional confusion matrix of binary classification tasks, these parameters represent the confusion matrix of multi-classification tasks.

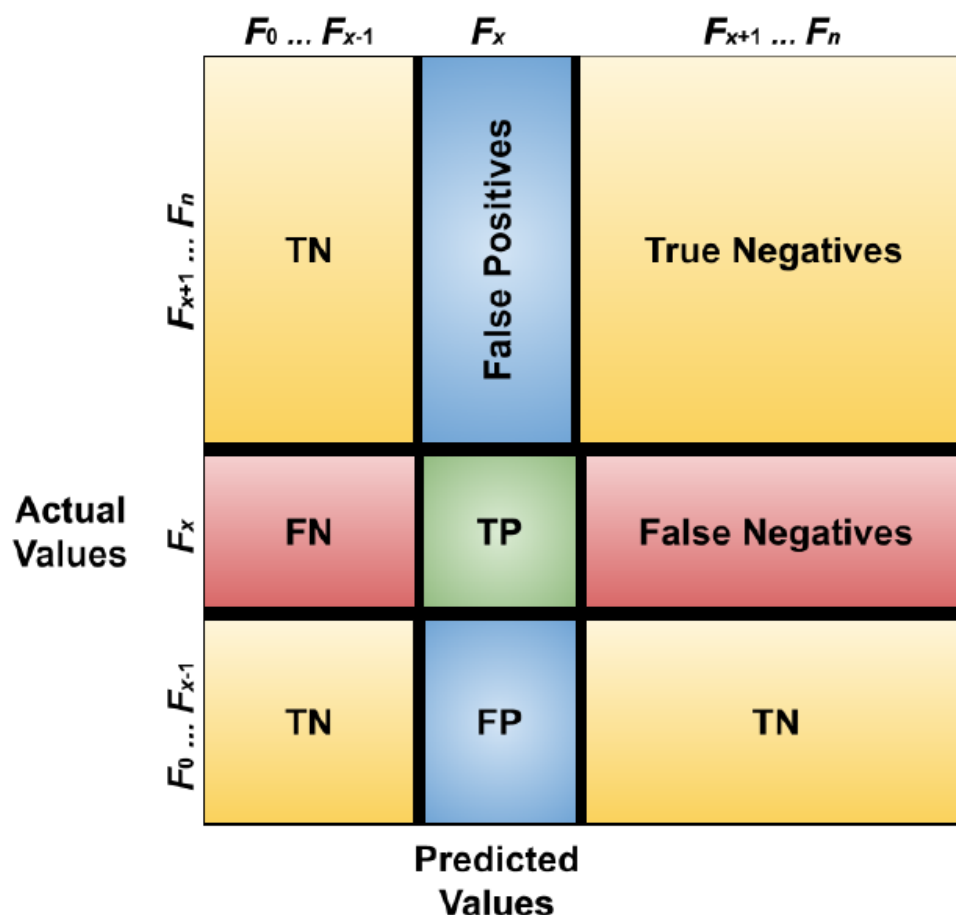


Figure 11. Multi-classification confusion matrix [18].

5.3. Evaluation Results

The model is assessed using the k-fold Cross-Validation process in order to produce accurate results. By randomly dividing the data into 70% training and 30% testing, the training and testing sets can be created. To guarantee that each malware family is fairly represented on the divided dataset, the proportional populations of each malware family are also taken into consideration when dividing the dataset. The results are classified into two parts: (a) straightforward features obtained from a manual features process (Tamura) and (b) deep features obtained from a pre-trained CNN model (GoogLeNet). Tables 1 and 2 show the overall accuracy of the proposed method based on three classifiers over the Maling dataset.

Table 2. The results of classification accuracy of three classifiers on Tamura feature across the Maling dataset.

Class ID	Family Name	Classification Accuracy (%)		
		KNN	SVM	ELM
#1	Adialer.C	100	98	100
#2	Agent.FYI	99	98	99
#3	Allaple.A	86	70	85
#4	Allaple.L	88	82	87
#5	Alueron.gen!J	99	98	99
#6	Autorun.K	100	99	100
#7	C2LOP.P	98	98	98
#8	C2LOP.gen!g	98	97	99
#9	Dialplatform.B	100	97	100
#10	Dontovo.A	100	99	100
#11	Fakerean	99	97	100
#12	Instantaccess	99	95	99
#13	Lolyda.AA1	99	97	99
#14	Lolyda.AA2	99	97	99
#15	Lolyda.AA3	100	100	100
#16	Lolyda.AT	99	97	99
#17	Malex.gen!J	99	98	99
#18	Obfuscator.AD	99	99	100
#19	Rbot!gen	99	99	99
#20	Skintrim.N	100	99	100
#21	Swizzor.gen!E	98	98	98
#22	Swizzor.gen!I	98	98	98
#23	VB.AT	99	96	99
#24	Wintrim.BX	99	98	99
#25	Yuner.A	100	96	100
	Average	95.34	93.23	95.42

5.3.1. Performance Results of the Proposed Based on Tamura

In the first experiment, three classifiers: KNN, SVM, and ELM, are employed for classification after applying the Tamura features approach to the Maling datasets for feature extraction. Table 2 displays the collected findings for each malware family and the total average accuracy metric following the application of each classifier.

Most malware families have accuracy levels of 90 or above across all classifiers. It is important to note that two malware families (Allaple.A and Allaple.L) have accuracy levels below 90, as depicted in Figure 12. Consequently, the outcomes are based on unbalanced features and data.

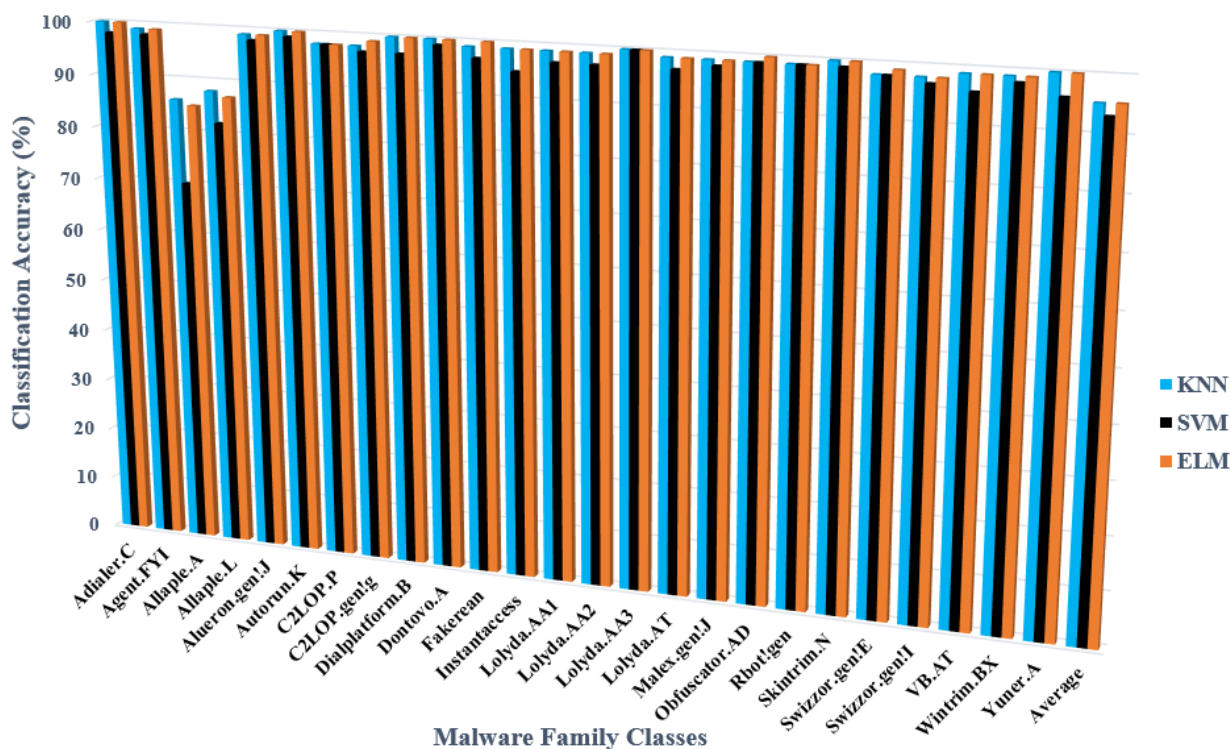


Figure 12. The classification accuracy based manual feature method across three classifiers.

The accuracy for each malware family, as seen in Table 2 and Figure 12, follows a similar pattern, especially for the four malware families (CL2OP.gen!g, C2LOP.P, Swizzor.gen!E, and Swizzor.gen!I). According to an examination of the picture data for each of the families, these malware families share comparable visual motifs, as depicted in Figure 13.

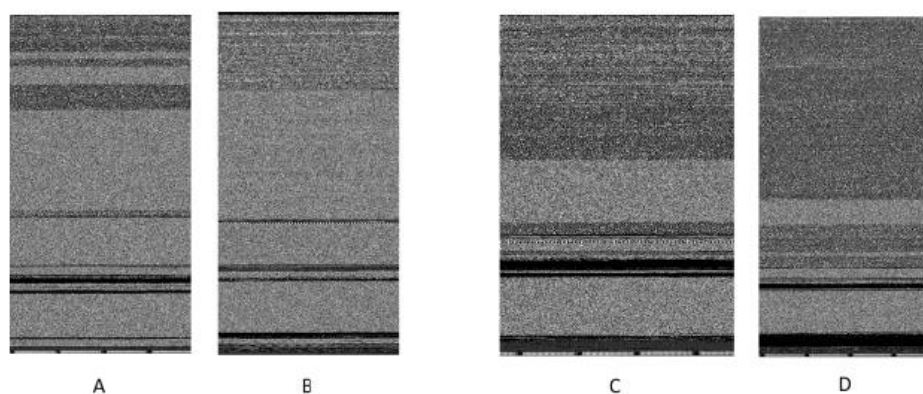


Figure 13. Graphical analysis of the malware families: (A) CL2OP.gen!g, (B) C2LOP.P, (C) Swizzor.gen!E, and (D) Swizzor.gen!I. showing matching visual patterns.

Even though all classifiers employ the same feature vector, their results vary. This is because each classifier has a unique set of traits. The graph in Figure 12 also includes the accuracy results of some classifiers. According to the accuracy results, the ELM on unbalanced datasets achieved a high classification accuracy of 95.42 percent using 10-fold cross-validation. The outcomes showed that the KNN produces substantial values (95.34) close to the ELM optimal. SVM performed poorly when compared to other classifiers, including KNN and ELM. For this purpose, in the first findings section of this paper, we took the proposed model based on Tamura and ELM classifier into consideration.

5.3.2. Performance Results of the Proposed Based on GoogLeNet

In the second experiment, three classifiers: KNN, SVM, and ELM, are employed for classification after applying the deep GoogLeNet features approach to the Maling datasets for feature extraction. Table 3 displays the collected findings for each malware family and the total average accuracy metric following the application of each classifier.

Table 3. The results of classification accuracy of three classifiers on deep feature across the Maling dataset.

Class ID	Family Name	Classification Accuracy (%)		
		KNN	SVM	ELM
#1	Adialer.C	98	99	98
#2	Agent.FYI	100	99	98
#3	Allaple.A	96	94	58
#4	Allaple.L	99	84	78
#5	Alueron.gen!J	100	97	96
#6	Autorun.K	100	98	97
#7	C2LOP.P	100	97	97
#8	C2LOP.gen!g	100	98	98
#9	Dialplatform.B	100	98	98
#10	Dontovo.A	100	98	98
#11	Fakerean	100	96	96
#12	Instantaccess	100	95	94
#13	Lolyda.AA1	100	98	98
#14	Lolyda.AA2	100	98	98
#15	Lolyda.AA3	100	99	98
#16	Lolyda.AT	100	99	99
#17	Malex.gen!J	100	99	97
#18	Obfuscator.AD	100	98	97
#19	Rbot!gen	100	98	97
#20	Skintrim.N	100	99	99
#21	Swizzor.gen!E	100	98	98
#22	Swizzor.gen!I	100	99	98
#23	VB.AT	100	95	95
#24	Wintrim.BX	100	98	99
#25	Yuner.A	100	98	90
	Average	96.84	94.24	92.07

In comparison to previous results, as shown in Table 2, the malware families with the highest number of variations, including Allaple.A and Allaple.L, have higher accuracy, especially using KNN and SVM Classifier, as shown in Table 3. This could be attributed to the dataset's separation into training data and testing data, excluding relative populations of each malware family. Most malware families have accuracy levels of 90 or above across all classifiers. It is important to note that two malware families (Allaple.A and Allaple.L) have accuracy levels below 80 as depicted in Figure 14, especially using the ELM Classifier. Consequently, the outcomes are based on unbalanced features and data.

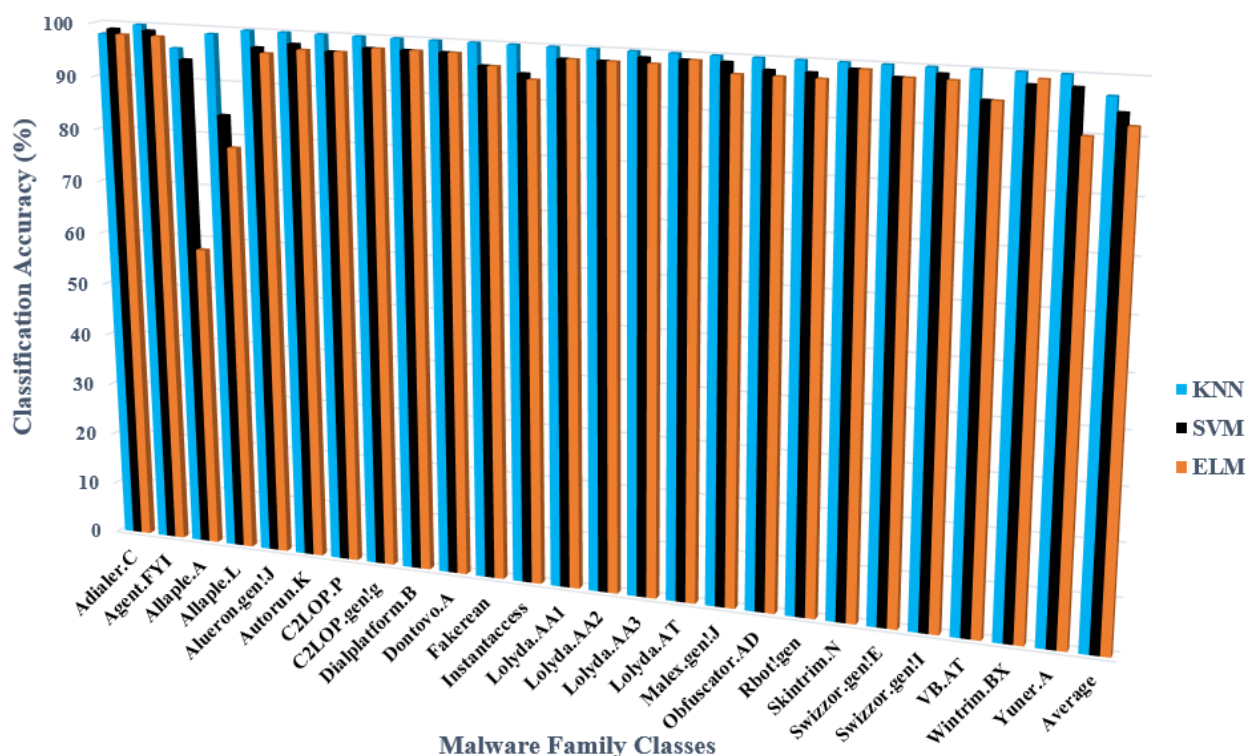


Figure 14. The classification accuracy based deep feature method across three classifiers.

According to the accuracy results, the KNN on an unbalanced datasets achieved a high classification accuracy of 96.84 percent using 10-fold cross-validation. KNN has the benefit of being very straightforward, not requiring a specific training phase, and being quite efficient for many applications, particularly when the training set is big. The outcomes showed that the SVM produces substantial values (94.24) close to the KNN optimal. ELM performed poorly when compared to other classifiers, including KNN and SVM. For this purpose, in the second findings section of this paper, we took the proposed model based on deep GoogLeNet features and the KNN classifier into consideration. This indicates that the performance results of the proposed methods are often consistent.

The results indicated that our method can efficiently and precisely classify malware to their relevant families even when the training datasets are unbalanced. While removing the human feature engineering phase, the proposed classification method exhibits good precision and completion time that is equivalent to traditional solutions based on machine learning.

The results show that the Google net CNN model’s deep features have a significant impact, especially when it comes to certain classes, such Allapple.A and Allapple.L. These classes’ overall accuracy in prior experiments were (86, 70, 85) & (88, 82, 87) and improved into (96, 94, 58) & (99, 84, 78). This demonstrates that the outcomes can be enhanced when these classes have more than 2000 instances. It is evident that the use of KNN and SVM classifiers significantly improves the outcomes.

These features are consistent and provide us with additional classification flexibility. With the help of this data set, we create an effective classification technique that uses a pre-trained network and a manually designed technique to decrease over training.

5.4. Comparison with Alternative Methods

To demonstrate the proposed method’s efficacy in identifying and classifying malware even when applied to identical datasets with unbalanced malware families, its performance is compared with a number of well-known methods that rely on hand-crafted features and deep Features. Most comparison techniques have been tested using the Maling datasets. In

terms of the accuracy metric, Table 4 displays the performance comparison of the proposed multi-classification framework employing Tamura, the GoogleNet CNN model, and three classifiers. Comparison of the proposed method to existing MC methods, which employ various hardware and software. Therefore, we use the outcomes of methods that employ the Malimg database, various feature domains, different texture feature extraction, and classification techniques.

Table 4. Comparison of the proposed method to existing MC methods which were evaluated on the Malimg dataset.

Feature Method Domain	Methods	Feature Extraction	Classifier Kind	Dataset	Accuracy (%)
Handcrafted Features	Garcia et al. [37]	Texture features	Random Forest	Malimg	95
	Dahl et al. [44]	Sparse binary features	Neural networks and logistic regression	Malimg	86
	Cui et al. [38]	GIST	SVM	Malimg	92.20
	Makandar & Patrot [8]	Gabor wavelet	KNN	Malimg	89.11
	Hsien-De et al. [45]	LBP	SVM	Malimg	75.02
	Proposed (Tamura + ELM)	Tamura	ELM	Malimg	95.42
Deep Features	Wen et al. [40]	static and dynamitic analysis	SVM	Malimg	95.2
	Manish and Raman, [39]	VGG-16, ResNet-50 and AlexNet	CNN	Malimg	98.88
	Yeo et al. [43]	CNN	MLP	Malimg	85
	Abttan et al. [42]	VGG16	Naive Bayes	Malimg	91.72
	Agarap et al. [12]	CNN	SVM	Malimg	84
	Rezende et al. [41]	VGG16	SVM	Malimg	92.29
	Khan et al. [19]	GoogleNet	GoogleNet	Malimg	74.5
	Abien & Agarap [12]	CNN	SVM	Malimg	77.22
Proposed (DEEP + KNN)	GoogleNet	KNN	Malimg	96.84	

After incorporating Tamura and ELM, the proposed classification accuracy increased to 95.42 percent, which is higher than that of conventional hand-crafted feature methods. The accuracy results of the methods [37,38] can be near to the presented method's optimal accuracy. However, these methods rely on GIST texture features, which demand huge feature vector dimensions. In contrast, our approach only needed a 1×3 feature vector. After incorporating GoogLeNet and KNN, the proposed classification accuracy increased to 96.94 percent, which is higher than that of conventional deep Feature methods.

The technique [39] may have the highest overall accuracy out of all the techniques. However, the combined feature of Pretrained VGG-16, ResNet-50, and AlexNet models involves greater computation time. The accuracy outcomes of the approaches [40–42] may be close to the best accuracy of the method described. As a result, the pre-trained feature of the VGG16 model displays impressive resilience. Unfortunately, processing these features took additional time. On the other hand, we attain high accuracy by exploiting the features of a pre-trained GoogleNet model. It is important to note that [19] is the least efficient method available. The usage of the end-to-end GoogleNet CNN model is the cause of this. Because of their whole reliance on full CNN, the techniques [12,43] give rather moderate results.

The proposed method’s accuracy rate was higher than both the hand-crafted feature and Deep Feature techniques, at 95.42 and 96.84 percent, respectively, as shown in Figures 15 and 16.

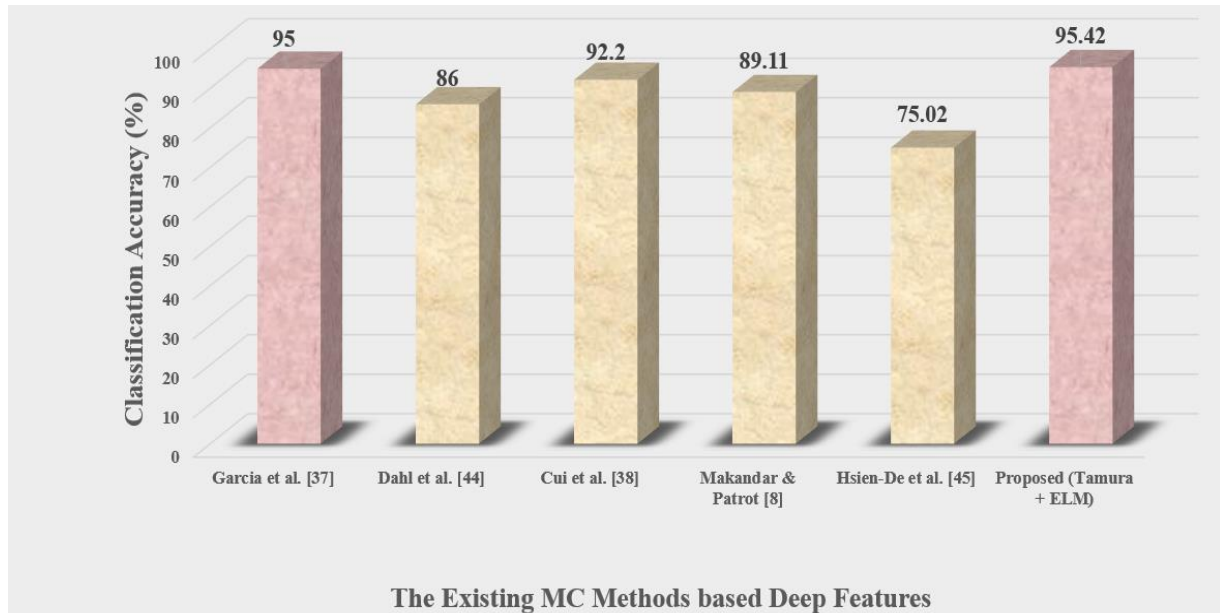


Figure 15. Accuracy results of different Hand-crafted Features methods comparison.

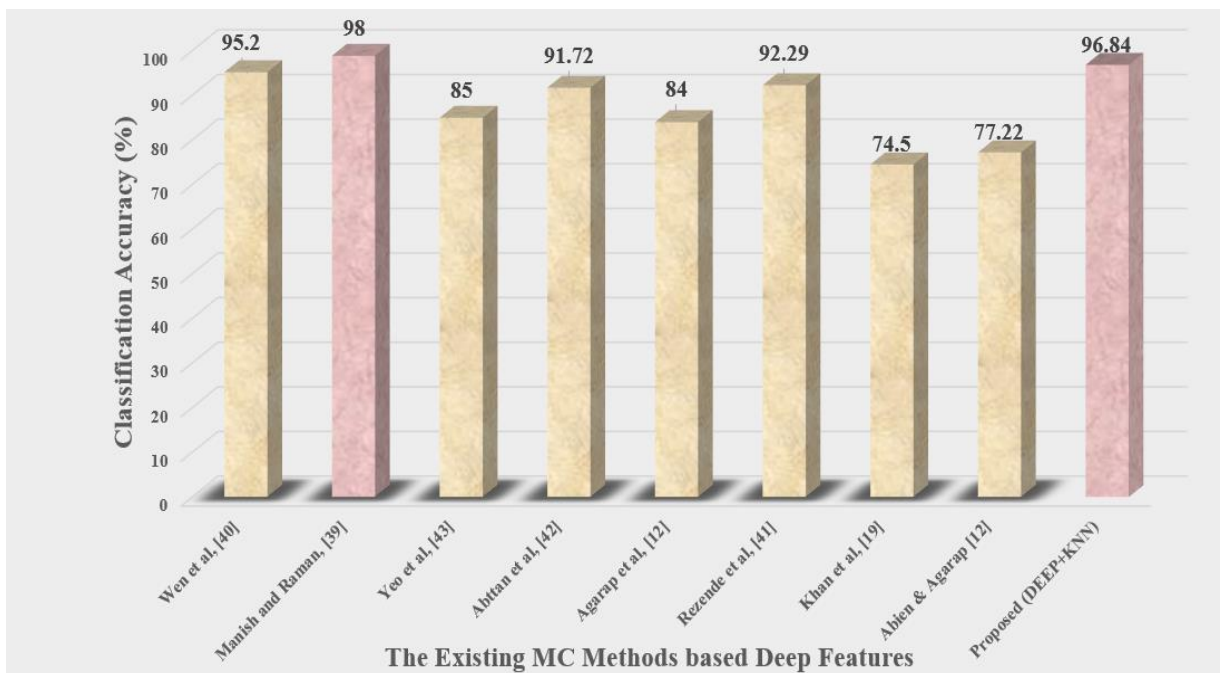


Figure 16. Accuracy results of different Deep Features methods comparison.

6. Conclusions

In order to improve classification performance while dealing with unbalanced datasets, the majority of conventional MC-based Techniques now in use employ a variety of data balancing techniques. In contrast, we classified malware images in this work with excellent accuracy and without using any data augmentation or other balancing strategies by using the unbalanced malware families of the benchmark Malimg database. The proposed

malware classification method consists of the following five steps: (i) Dataset preparation: 2D malware images are created from the malware binary files; (ii) Visualized Malware Pre-processing: the visual malware images need to be scaled to fit the CNN model's input size; (iii) Feature extraction: both hand-engineering (Tamura) and deep learning (GoogLeNet) techniques are used to extract the features in this step; (iv) classification: to perform malware classification, we employed KNN, SVM, and ELM. After incorporating GoogLeNet and KNN, the proposed classification accuracy increased to 96.94 percent, which is higher than that of conventional deep Feature methods. The proposed method's accuracy rate was higher than both the hand-crafted feature and Deep Feature techniques, at 95.42 and 96.84 percent, respectively. This indicates that the performance results of the proposed methods are often consistent. An important factor in improving malware classification findings is data augmentation. Therefore, data augmentation may be applied in the future. Additionally, we investigated different feature engineering domains.

Author Contributions: Conceptualization, I.T.A., B.T.H. and N.J.; writing—original draft preparation, I.T.A.; writing—Original Draft Preparation, B.T.H.; review & editing, N.J.; Validation, Z.M.Z.; Software, I.T.A.; Validation, S.B.; Funding acquisition, S.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R195), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: This research is supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2022R195), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

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

References

1. Berman, D.S.; Buczak, A.L.; Chavis, J.S.; Corbett, C.L. A survey of deep learning methods for cyber security. *Information* **2019**, *10*, 122. [[CrossRef](#)]
2. Hemalatha, J.; Roseline, S.A.; Geetha, S.; Kadry, S.; Damaševičius, R. An efficient densenet-based deep learning model for malware detection. *Entropy* **2021**, *23*, 344. [[CrossRef](#)] [[PubMed](#)]
3. Poudyal, S.; Akhtar, Z.; Dasgupta, D.; Gupta, K.D. Malware analytics: Review of data mining, machine learning and big data perspectives. In Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6–9 December 2019; pp. 649–656.
4. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburgh, PA, USA, 20 July 2011; pp. 1–7.
5. Barath, N.N.; Ouboti, D.B.; Temesguen, M.K. Pattern recognition algorithms for malware classification. In Proceedings of the 2016 IEEE conference of aerospace and electronics, Dayton, OH, USA, 5–12 March 2016; pp. 338–342.
6. Kosmidis, K.; Kalloniatis, C. Machine learning and images for malware detection and classification. In Proceedings of the 21st Pan-Hellenic Conference on Informatics, Larissa, Greece, 26–28 November 2017; pp. 1–6.
7. Naeem, H.; Guo, B.; Naeem, M.R.; Vasani, D. Visual malware classification using local and global malicious pattern. *J. Comput.* **2019**, *6*, 73–83.
8. Makandar, A.; Patrot, A. Malware class recognition using image processing techniques. In Proceedings of the 2017 International Conference on Data Management, Analytics and Innovation (ICDMAI), Pune, India, 24–26 February 2017; pp. 76–80.
9. Verma, V.; Muttoo, S.K.; Singh, V.B. Multiclass malware classification via first-and second-order texture statistics. *Comput. Secur.* **2020**, *97*, 101895. [[CrossRef](#)]
10. Sun, G.; Qian, Q. Deep learning and visualization for identifying malware families. *IEEE Trans. Dependable Secur. Comput.* **2018**, *18*, 283–295. [[CrossRef](#)]
11. Gibert, D.; Mateu, C.; Planes, J.; Vicens, R. Using convolutional neural networks for classification of malware represented as images. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 15–28. [[CrossRef](#)]
12. Agarap, A.F. Towards building an intelligent anti-malware system: A deep learning approach using support vector machine (SVM) for malware classification. *arXiv* **2017**, arXiv:1801.00318.

13. Çayır, A.; Ünal, U.; Daug, H. Random CapsNet forest model for imbalanced malware type classification task. *Comput. Secur.* **2021**, *102*, 102133. [[CrossRef](#)]
14. Gibert, D. *Convolutional Neural Networks for Malware Classification*; University Rovira i Virgili: Tarragona, Spain, 2016.
15. David, O.E.; Netanyahu, N.S. DeepSign: Deep learning for automatic malware signature generation and classification. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–8.
16. Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševičius, R.; Blažauskas, T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. *Appl. Sci.* **2020**, *10*, 4966. [[CrossRef](#)]
17. 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](#)]
18. El-Shafai, W.; Almomani, I.; AlKhayer, A. Visualized malware multi-classification framework using fine-tuned CNN-based transfer learning models. *Appl. Sci.* **2021**, *11*, 6446. [[CrossRef](#)]
19. Khan, R.U.; Zhang, X.; Kumar, R. Analysis of ResNet and GoogleNet models for malware detection. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 29–37. [[CrossRef](#)]
20. Bennasar, H.; Bendahmane, A.; Essaïdi, M. An overview of the state-of-the-art of cloud computing cyber-security. In Proceedings of the International Conference on Codes, Cryptology, and Information Security, Rabat, Morocco, 10–12 April 2017; pp. 56–67.
21. Roseline, S.A.; Sasisri, A.D.; Geetha, S.; Balasubramanian, C. Towards efficient malware detection and classification using multilayered random forest ensemble technique. In Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST), Kota, Kinabalu, 1–3 October 2019; pp. 1–6.
22. Ben Abdel Ouahab, I.; Bouhorma, M.; Boudhir, A.A.; El Aachak, L. Classification of grayscale malware images using the K-nearest neighbor algorithm. In Proceedings of the the Third International Conference on Smart City Applications, Karabuk, Turkey, 7–9 October 2019; pp. 1038–1050.
23. Awan, M.J.; Masood, O.A.; Mohammed, M.A.; Yasin, A.; Zain, A.M.; Damaševičius, R.; Abdulkareem, K.H. Image-Based Malware Classification Using VGG19 Network and Spatial Convolutional Attention. *Electronics* **2021**, *10*, 2444. [[CrossRef](#)]
24. Kumar, S.; Sudhakar. MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things. *Futur. Gener. Comput. Syst.* **2021**, *125*, 334–351.
25. Xiao, G.; Li, J.; Chen, Y.; Li, K. MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks. *J. Parallel Distrib. Comput.* **2020**, *141*, 49–58. [[CrossRef](#)]
26. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
27. 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.
28. Khan, S.H.; Sohail, A.; Khan, A.; Lee, Y.S. Classification and region analysis of COVID-19 infection using lung CT images and deep convolutional neural networks. *arXiv* **2020**, arXiv:2009.08864.
29. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 315–323.
30. Tanveer, M.; Shubham, K.; Aldhaifallah, M.; Ho, S.S. An efficient regularized K-nearest neighbor based weighted twin support vector regression. *Knowl. Based Syst.* **2016**, *94*, 70–87. [[CrossRef](#)]
31. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4.
32. Ahmed, I.T.; Hammad, B.T.; Jamil, N. Image Copy-Move Forgery Detection Algorithms Based on Spatial Feature Domain. In Proceedings of the 2021 IEEE 17th International Colloquium on Signal Processing & Its Applications (CSPA), Langkawi, Malaysia, 5–6 March 2021; pp. 92–96.
33. Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme learning machine: A new learning scheme of feedforward neural networks. In Proceedings of the 2004 IEEE International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 18–21 November 2004; Volume 2, pp. 985–990.
34. Hoang, N.-D.; Bui, D.T. Slope stability evaluation using radial basis function neural network, least squares support vector machines, and extreme learning machine. In *Handbook of Neural Computation*; Elsevier: Amsterdam, The Netherlands, 2017; pp. 333–344.
35. Jain, M.; Andreopoulos, W.; Stamp, M. CNN vs ELM for Image-Based Malware Classification. *arXiv* **2021**, arXiv:2103.13820.
36. Ahmed, I.T.; Hammad, B.T.; Jamil, N. A comparative analysis of image copy-move forgery detection algorithms based on hand and machine-crafted features. *Indones. J. Electr. Eng. Comput. Sci.* **2021**, *22*, 1177–1190. [[CrossRef](#)]
37. Garcia, F.C.C.; Muga II, F.P. Random forest for malware classification. *arXiv* **2016**, arXiv:1609.07770.
38. 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](#)]
39. Goyal, M.; Kumar, R. AVMCT: API Calls Visualization based Malware Classification using Transfer Learning. *J. Algebraic Stat.* **2022**, *17*, 31–41.
40. Wen, L.; Yu, H. An Android malware detection system based on machine learning. In Proceedings of the AIP Conference Proceedings, Tokyo, Japan, 1–2 November 2017; Volume 1864, p. 20136.
41. Rezende, E.; Ruppert, G.; Carvalho, T.; Theophilo, A.; Ramos, F.; de Geus, P. Malicious software classification using VGG16 deep neural network's bottleneck features. In *Information Technology-New Generations*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 51–59.
42. Choudhary, S.; Sharma, A. Malware detection & classification using machine learning. In Proceedings of the 2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3), Sikar, India, 21–22 February 2020; pp. 1–4.

43. Yeo, M.; Koo, Y.; Yoon, Y.; Hwang, T.; Ryu, J.; Song, J.; Park, C. Flow-based malware detection using convolutional neural network. In Proceedings of the 2018 International Conference on Information Networking (ICOIN), Chiang Mai, Thailand, 10–12 January 2018; pp. 910–913.
44. 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, Vancouver, UK, 26–31 May 2013; pp. 3422–3426.
45. Hsien-De Huang, T.; Kao, H.-Y. R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 2633–2642.