applied sciences

Article

Visualized Malware Multi-Classification Framework Using Fine-Tuned CNN-Based Transfer Learning Models

Walid El-Shafai [1,2], Iman Almomani [1,3,*] and Aala AlKhayer [1]

1 Security Engineering Lab, Computer Science Department, Prince Sultan University, Riyadh 11586, Saudi Arabia; welshafai@psu.edu.sa or walid.elshafai@el-eng.menofia.edu.eg (W.E.-S.); akhayer@psu.edu.sa (A.A.)
2 Department of Electronics and Electrical Communications Engineering, Faculty of Electronic Engineering, Menoufia University, Menouf 32952, Egypt
3 Computer Science Department, King Abdullah II School for Information Technology, The University of Jordan, Amman 11942, Jordan
* Correspondence: imomani@psu.edu.sa or i.momani@ju.edu.jo

Abstract: There is a massive growth in malicious software (Malware) development, which causes substantial security threats to individuals and organizations. Cybersecurity researchers makes continuous efforts to defend against these malware risks. This research aims to exploit the significant advantages of Transfer Learning (TL) and Fine-Tuning (FT) methods to introduce efficient malware detection in the context of imbalanced families without the need to apply complex features extraction or data augmentation processes. Therefore, this paper proposes a visualized malware multi-classification framework to avoid false positives and imbalanced datasets' challenges through using the fine-tuned convolutional neural network (CNN)-based TL models. The proposed framework comprises eight different FT CNN models including VGG16, AlexNet, DarkNet-53, DenseNet-201, Inception-V3, Places365-GoogleNet, ResNet-50, and MobileNet-V2. First, the binary files of different malware families were transformed into 2D images and then forwarded to the FT CNN models to detect and classify the malware families. The detection and classification performance was examined on a benchmark Malimg imbalanced dataset using different, comprehensive evaluation metrics. The evaluation results prove the FT CNN models' significance in detecting malware types with high accuracy that reached 99.97% which also outperforms the performance of related machine learning (ML) and deep learning (DL)-based malware multi-classification approaches tested on the same malware dataset.

Keywords: cybersecurity threats; malware visualization; detection; classification; deep learning; machine learning; CNN; transfer learning; fine-tuning; VGG16

Citation: 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. https://doi.org/10.3390/app11146446

Academic Editors: Luis Javier Garcia Villalba, Ana Lucila Sandoval Orozco, Rafael T. de Sousa, Jr. and Robson de Oliveira Albuquerque

Received: 23 May 2021
Accepted: 8 July 2021
Published: 13 July 2021

1. Introduction

Malware, short for Malicious Software, is a compiled binary file that interrupts computer systems or networks aiming to steal data, modify/delete or encrypt sensitive information, and hijack core computing functions. Malware includes several types such as worms, trojans, spyware, and ransomware [1]. Recently, there has been a tremendous increase in malware development by an average of 588 cyber threats per minute according to McAfee report 2021 [2]. Consequently, Malware classification and detection have become one of the most important research fields. Malware detection is mainly performed by implementing static-based analysis or dynamic-based analysis [3,4]. In the static-based analysis, the original source code of the malware is statically scanned without executing the code. Even though this approach is inexpensive, it is insufficient in the case of encrypted or obfuscated malware attacks. However, the dynamic-based approach analyzes the behavioral features of the malicious software by executing the code in isolated or virtual environments. Hence, this approach consumes time and resources.

Another alternative to malware classification is the visualization approach. Many recent works have been used the malware visualization analysis approach [5–7] as an efficient solution for malicious software classification since it analyzes the malware executable file as a whole. Malware visualization is a method in which malicious software is converted into an image by extracting its binaries [8]. Since the malicious code is visualized, each malicious family presents a special texture pattern of the generated images of the malware applications that belong to the same family. An additional advantage of the malware visualization analysis that it does not require static decompilation or dynamic running of malware software. After the malware visualization, the training classifier could be implemented by deploying the texture features of the malware image. Consequently, even if the attacker has employed obfuscation or modification techniques, the texture representing the malicious software will be exhibited in the malware image [5].

One of the most prominent neural network models is Convolutional Neural Networks (CNNs) which are used in image classification. CNN provides a superior data representation, thus features engineering can be avoided. Initially, the input image is converted into an array of pixels. Subsequently, the image is processed by several convolutional layers to finally generate a predicted output [9]. To train the CNN models, huge and well-illustrated datasets are used, such as ImageNet [10]. However, to enhance the malware classification and detection, the gained knowledge out of CNNs can be transferred to a different learning task [11]. The core advantage of transfer learning is that it enables training a task with a limited dataset by using a pre-trained model with a large-scale dataset.

This research proposes a vision-based malware multi-classification framework that aims to overcome the shortcomings of the existing malware detection mechanisms. The proposed framework recruits the advantages of CNN models trained on large-scale datasets by transferring their knowledge to fine-tuning phase to improve the detection accuracy without building the training models from scratch. Moreover, the proposed framework does not need to run expensive processes applied by conventional ML and DL techniques, including features extraction and data augmentation to balance the malware datasets under study.

Therefore, the main contributions of this research work can be summarized as follows:

- Developing eight different fine-tuned CNN-based transfer learning (TL) models for vision-based malware multi-classification applications.
- Using CNN models for vision-based malware classification, which do not require features engineering such as binary disassembly or reverse engineering to detect visual malware samples.
- Employing fine-tuned CNN models to function properly on 9341 images of 25 different malware families of 8 malware types.
- Achieving high classification accuracy with fewer epochs and iterations for the developed CNN models than recent related work.
- Succeeding to develop CNN models that can efficiently perform malware detection on imbalanced datasets (e.g., Malimg dataset [12]).
- Applying extensive performance analysis in terms of 15 different evaluation metrics to assess the examined fine-tuned CNN models accurately.
- Conducting in-depth comparative analysis among the employed fine-tuned models and the recent related ML and DL models in terms of the obtained classification accuracy and other detection metrics.

The rest of this paper is structured as follows. Section 2 presents a comprehensive summary of the recent related malware-based detection approaches. The proposed visualized malware multi-classification framework is explained in Section 3. The simulation results and comparisons are discussed in Section 4. Concluding remarks and some future directions are presented in Section 5.

2. Literature Survey

Many malware classification and identification research works have been investigated based on different analysis approaches such as static-based, dynamic-based, and machine learning-based [13–16]. This section provides a comprehensive survey of several malware classification methods. Static-based analysis deploys functional call-graph [17], features of portable executable (PE) malware files [18], function length frequencies [19], and opcode sequences [20]. Almomani et al. [21–23] implemented a static analysis to extract several static features from Android malware binary files such as permissions and API calls. Subsequently, they performed machine learning techniques to detect malware applications [24]. The authors of [25] developed a static analysis with Tensorflow (SAT) malware detection system. The proposed system performs a static analysis by employing a signature-based method on both known and new/modified malware. However, static-based analysis is not sufficient in the case of code obfuscation and zero-day malware [5].

In the dynamic-based analysis, the behavioral characteristics of malware are obtained, such as API calls [26], network activities [27], and log files [28]. In [27], Mohaisen et al. developed an automated malware and labeling scheme (AMAL). In the proposed system, several behavior-based features were extracted during the dynamic analysis, such as network activities, file systems, and registers. The authors of [28] integrated memory forensics techniques with a dynamic analysis approach. Initially, malicious artifacts were extracted from the memory. Subsequently, the Cuckoo Sandbox was deployed to monitor the malware behavior during its execution. Finally, the malicious artifacts and the behavioral report were combined to create the features dataset for further classification. However, malicious software may alter its behavior during its execution in a virtual environment; thus, the dynamic-based analysis might fail to capture the actual behavior of the malicious software.

Recently, extensive research on malware classification has been made by deploying the vision-based approach [6–8,29–31]. Some authors developed CNN solutions from scratch in which they did not use any pre-trained models [7,9,29,30]. In [7], the authors developed a visualized malware classification system based on Artificial Neural Network (ANN). The proposed classification system used the extracted features of the Malimg database to train ANN. Subsequently, the trained model is further employed in classifying different samples of the Malimg database. The accuracy of applying one hidden layer was 96%. However, implementing two hidden layers achieved an accuracy of 99.135%. Gibert et al. have also visualized malware as gray-scale images to develop a file agnostic deep learning scheme based on CNN [29]. The proposed scheme extract patterns to classify malicious software allowing the malicious software to be classified in a real-time environment. Besides patterns, different features could be deployed in the malware visualization process. In [30], they have included the local features in visualizing the malware application by using the FastText model. Consequently, each malware family has a unique generated local malware image since the proposed system mainly includes the local features of each malicious software.

Furthermore, the authors of [32] proved that combining deep CNN with an entropy graph contributes to enhancing the malware pattern classification process. In [33], the authors have investigated the classification on a different color scale in which they converted the malware APK files into colored images instead of gray-scale images. However, they have compared applying the classification between gray-scale and colored images and proved that applying the classification on colored images outperformed the gray-scale classification. Vasan et al. have also deployed colored images in which they have implemented IMCFN, an image-based malware classification system using Fine-tuned CNN [31]. Initially, they have converted malware into colored images using a colored map algorithm. To overcome the imbalanced dataset issue, they applied data augmentation during the fine-tuning process. Furthermore, they have also compared applying the classification between gray-scale and colored images and achieved the same result of having better classification performance in the case of using colored images. Even though IMCFN accomplished an accuracy of 98.82% in the Malimg dataset, the proposed system bears an extra complexity due to the used augmentation techniques and colored map algorithm.

Some researchers might choose to combine several training models to improve the classification process [8,34–37]. In [8], the SFTA (scale feature texture analyzer) is combined with two models of deep CNN (DCNN), AlexNet and Inception-v3 techniques, to enhance the accuracy of malware detection. Another combination of CNN models was proposed by [35] in which the authors have uses VGG16 and ResNet50 for features extraction. However, in the ensemble of CNN architectures, two classifiers have been deployed SoftMax and Multiclass SVMs. Following that, a PCA (principal component analysis) process was applied to decrease the dimensionality of the features, while a fusion process was used before the classification process. Moreover, the CNNs might be combined with other techniques such as Long Short-Term Memory (LSTM) [36]. The proposed solution by [36] implemented an ensemble classification scheme based on recurrent and convolutional neural networks by using the complied and the assembled malware files. They have classified the visualized image of the complied malware files using CNN while the LSTM was used to classify Assembly files of malicious software.

In other scenarios, the CNN might be applied to extract features while machine learning techniques are deployed in the malware classification process [38–40]. In [38], the visualized malware was classified by deploying a sequential multilayered Random Forest ensemble technique. The suggested solution is performed in two stages. Initially, the raw features were analyzed using different sizes of sliding windows. Subsequently, four different machine learning (ML) techniques were applied including, Random forests (RF), Xgboost, Extra trees classifier (ETC), and Logistic regression (LR). The authors of [39] developed an image feature descriptor to extract the similarities among the malware images. Then, they have deployed the k-Nearest Neighbor (KNN) algorithm to perform the classification process. Another machine learning-based classification system was proposed by [40] in which they deployed the local and global malicious patterns (LGMP) to extract the features of the visualized malware.

Table 1 presents a comprehensive comparison among the most recent related studies that have deployed the imbalanced Malimg dataset in their proposed systems. After observing the limitations of state-of-the-art malware classification approaches, different fine-tuned CNN-based TL architectures are introduced in this paper to significantly reduce the misclassification rate without increasing the complexity. Thus, different from the prior malware detection approaches, this work implemented eight CNN models for visualized malware multi-classification purposes. First, the PE (portable executable) malware samples are converted to gray-scale images to build the malware dataset in a proper format that suits the input type of the developed CNN models. An image processing stage is then introduced to the obtained malware images to resize them appropriately to meet the input size conditions of the used CNN models. After that, the fine-tuning process is performed for the pre-trained CNN-based TL models that were trained on ImageNet database. Consequently, avoiding the expected misclassification in testing the imbalanced Malimg dataset by transferring the obtained optimum weights of the pre-trained CNN-based TL models to the malware classification tasks. Subsequently, these developed fine-tuned CNN-based TL models are utilized to classify 25 malware families of the imbalanced Malimg dataset. The fine-tuning process of the CNN layers and hyperparameter values assists in identifying different malicious software families and enhancing the pre-trained models' classification performance without employing data augmentation techniques.

Table 1. Comparison among the recent related works that have employed Malimg in their studies.

Work	The Aim	Classification Approach	Model	Input	Accuracy
[5]	To propose a malware detection solution that handles imbalanced data	Deep learning	DenseNet	gray-scale images	97.55%
[6]	To develop a deep forest model by implementing a layered ensemble approach	Deep Random Forest	Sliding Window	gray-scale images	98.65%
[7]	To develop a visualized malware classification system based on ANN	ANN	Scratch model	file properties, gray-scale image	99.135%
[8]	To improve the accuracy of malware detection by combining AlexNet and Inception-v3 techniques	DCNN, SFTA	AlexNet	gray-scale images	99.3%
[9]	To implement a lightweight CNN to classify malware images	Lightweight CNN	Scratch model	gray-scale images	97.68%
[29]	To develop a file agnostic deep learning scheme to effectively group malware into families based on their patterns	CNN	Scratch model	gray-scale images	98.4%
[31]	To implement a CNN classification system based on colored images of malware	CNN	VGG16, ResNet50	colored images	98.82%
[34]	To develop an accurate and fast classification system by integrating CNN with Gradient Boosting algorithm	CNN	XG-Boost	gray-scale images	98.7%
[32]	To combine deep CNN with an entropy graph to enhance the malware pattern classification process	CNN	Scratch model	gray-scale images	99.67%
[33]	To implement an industrial Internet of things malware classification system based on colored image visualization	DCNN	Scratch model	colored images	98.79%
[35]	To implement a visualization-based malware detection system using VGG16 and ResNet50 CNN models	CNN	VGG16, ResNet50	gray-scale images	98.11%
[38]	To perform malware classification by deploying a sequential multilayered Random Forest ensemble technique	ML	RF, Xgboost, ETC, LR	gray-scale images	98.91%
[39]	To classify malware images using an image feature descriptor to extract the similarities among them	ML	KNN	gray-scale images	97%
[40]	To deploy the local and global malicious patterns in the malware classification process	ML	LGMP	gray-scale images	98%

3. Proposed Visualized Malware Multi-Classification Framework

Effective detection of windows malware families is a mandatory aspect in Internet security applications. Rather than employing the conventional features extraction-based ML approaches that require high computational processing in texture analysis, in this paper, efficient visualized malware multi-classification models are introduced based on deep learning (DL) approaches. Therefore, to efficiently identify malware images with reduced computations and achieve maximum identification accuracy, different fine-tuned CNN-based TL models are developed and used, as shown in the proposed malware multi-classification framework in Figure 1.

The used fine-tuned CNN models do not use reverse engineering for the malware multi-classification process. The basic procedure of the proposed malware multi-classification framework is shown in Figure 1. It consists of five different phases: (1) Dataset preparation, (2) Pre-processing, (3) Transfer learning, (4) Fine-tuning and classification, and (5) Performance evaluation. The details of these phases are as follows:

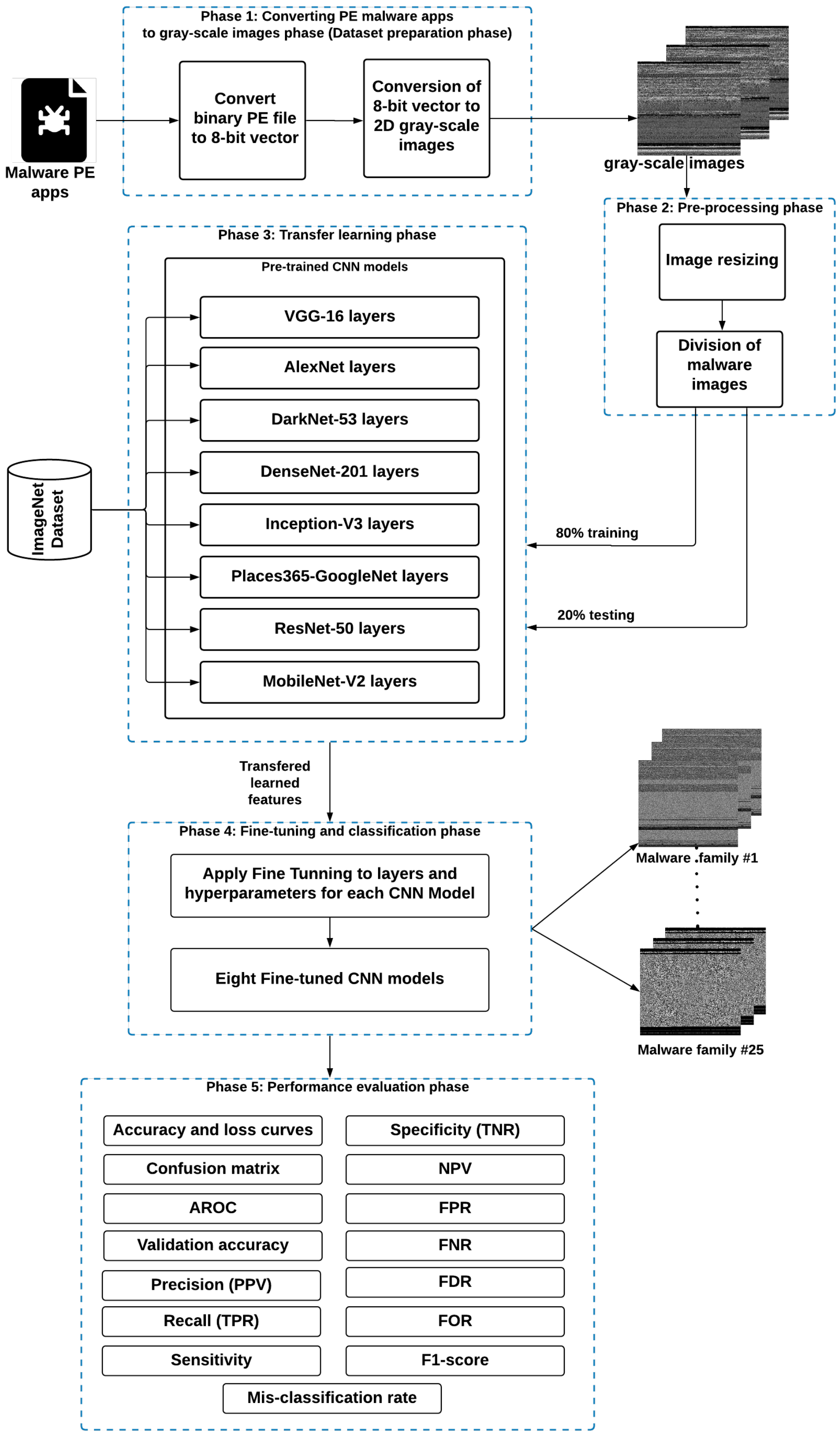

Phase 1: Converting PE malware apps to gray-scale images phase (Dataset preparation phase)
Malware PE apps
Convert binary PE file to 8-bit vector
Conversion of 8-bit vector to 2D gray-scale images
gray-scale images
Phase 2: Pre-processing phase
Image resizing
Division of malware images
80% training
20% testing
Phase 3: Transfer learning phase
Pre-trained CNN models
VGG-16 layers
AlexNet layers
DarkNet-53 layers
DenseNet-201 layers
Inception-V3 layers
Places365-GoogleNet layers
ResNet-50 layers
MobileNet-V2 layers
ImageNet Dataset
Transfered learned features
Phase 4: Fine-tuning and classification phase
Apply Fine Tunning to layers and hyperparameters for each CNN Model
Eight Fine-tuned CNN models
Malware family #1
Malware family #25
Phase 5: Performance evaluation phase
Accuracy and loss curves
Specificity (TNR)
Confusion matrix
NPV
AROC
FPR
Validation accuracy
FNR
Precision (PPV)
FDR
Recall (TPR)
FOR
Sensitivity
F1-score
Mis-classification rate

Figure 1. Proposed visualized malware multi-classification framework.

3.1. Dataset Preparation Phase

This phase is concerned with converting PE (portable executable) malware apps to gray-scale images. A PE malware binary is converted to a visual image to obtain and extract the main features and details of the malware apps. As shown in Figure 1, the malware PE application is first converted to a 1D 8-bit binary vector (unsigned integers). Then, this obtained 8-bit vector is transformed into a visual 2D gray-scale image. The main advantage of converting PE malware app to visual malware image is that it does not necessitate any domain expertise or feature engineering knowledge. Table 2 illustrates some samples of the visualized malware images after rearranging the 1D bit vectors of the malware binaries into 2D visual arrays. It can be observed from these malware images that the image width is variant for each malware family, which depends on the malware app size. Table 3 presents different widths for the malware images due to different sizes of malware files. Additionally, we can conclude from Table 2 that the obtained visual images of a variety of malware families exhibited differently in style, layout, and form. Therefore, each malware family has its own visual characteristics and similarities that are different from other malware families, where each family has various visualization features and distinct stripes. Such observations have motivated this research to adapt and tune the general CNN algorithms used for digital image classification into malware detection tasks.

Table 2. Samples of 2D visualized malware images.

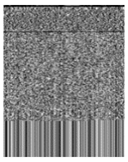
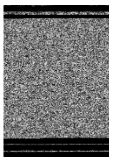
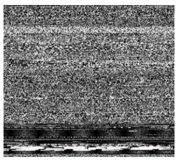
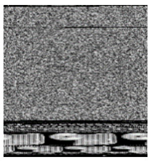
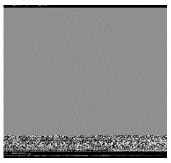
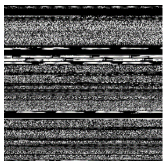
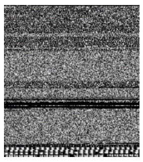
Adialer_C	Agent_FYI	 Allaple_A	 Alueron_gen!J
 Autorun_K	Dialplatform_B	Dontovo_A	 Fakerean
Rbot!gen	 Lolyda_AA3	 VB_AT	 Swizzor_gen!E

Table 3. Various malware image widths for different malware files' sizes.

Malware File Size	Malware Image Width
>1000 KB	1024
500∼1000 KB	768
200∼500 KB	512
100∼200 KB	384
60∼100 KB	256
30∼60 KB	128
10∼30 KB	64
<10 KB	32

3.2. Pre-Processing Phase

As discussed in the first phase, the malware binaries are converted to 2D malware images with different sizes that are not fixed among images of the tested 25 families.

Thus, a pre-processing step for malware data are a mandatory stage to be introduced to reformate the input image size corresponding to the CNN algorithms' settings. Therefore, the objective of this phase is to resize the malware images obtained in the first phase to an appropriate size to be compatible with the input size of the employed CNN model, where each fine-tuned CNN model from the eight examined models has its own standard size for the input image, as shown in Table 4. The foremost advantage of the resizing process is reducing the input image sizes, which is very beneficial in accelerating the training process and decreasing the computational overhead of the employed CNN model. Moreover, the main texture features of the malware images are preserved during the re-dimensionality process.

Furthermore, in this phase, the malware images dataset is divided into two different ratios for training and testing purposes. In this work, several simulation experiments were conducted to choose the proper ratios regarding detection accuracy and execution performance. The experiments' results revealed that allocating 80% of the malware samples for training and 20% for testing have achieved the superior and recommended malware detection accuracy compared to the other training and testing ratios for the examined CNN models. Both 20% and 80% of the samples were selected randomly by our proposed framework.

Table 4. The tested fine-tuned CNN models and their input size.

No.	Tested CNN Model	Input Image Size
1	VGG16	$224 \times 224 \times 3$
2	AlexNet	$227 \times 227 \times 3$
3	DarkNet-53	$256 \times 256 \times 3$
4	DenseNet-201	$224 \times 224 \times 3$
5	Inception-V3	$299 \times 299 \times 3$
6	Places365-GoogleNet	$224 \times 224 \times 3$
7	ResNet-50	$224 \times 224 \times 3$
8	MobileNet-V2	$224 \times 224 \times 3$

3.3. *Transfer Learning (TL) Phase*

TL refers to transferring the CNN parameters of a specified detection task with a specific image database to a new classification challenge with a different detection task for another image database. Almost all deep CNN models trained and learned on natural digital images have a common phenomenon: they understand and discover the general features of the input images through their first CNN layers, where these features are not specific to a particular task or dataset. However, they can be applied to many classification tasks and different image datasets. Therefore, the benefit of TL can be a formidable solution when the target database is considerably smaller than the original database; this is to avoid the overfitting occurrence, especially in the case of imbalanced datasets.

Consequently, the malware classification task can be considered to be an image classification task, especially when the malware binary samples are converted into visual malware images. Thus, the standard CNN models used for natural image classification can be exploited to classify the visualized malware images. In this regard, the TL-based CNN models trained on the benchmark ImageNet database [41] can be efficiently adapted to detect malware families. This database is updated through an annual competition called the "ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)" which is created for visual object detection challenges.

Many CNN models were already trained on natural images such as VGG16 [42], AlexNet [43], DarkNet-53 [44], DenseNet-201 [45], Inception-V3 [46], Places365-GoogleNet [47], ResNet-50 [48], and MobileNet-V2 [49]. In this paper, to obtain and extract the main features of the malware images, we used these eight pre-trained CNN models on the ImageNet database to identify general objects. Therefore, these models can be retrained and tested quickly using malware images to extract the main features from the input malware images; this is the great benefit of the TL concept. Thus, TL-based CNN architectures have started recently to be employed for intrusion detection and malware classification research. This is because TL can offer effective and promising detection solutions through knowledge transfer from standard image detection tasks to malware image detection tasks.

Among all TL-based CNN models examined in this paper, the fine-tuned VGG16 model accomplishes the best superior and promising results for visual malware multi-classification compared to other models. Therefore, we provide deep insights into its architecture, behavior, parameters, and simulation results. Thus, the proposed multi-classification framework has implemented the fine-tuned version of the pre-trained VGG16 CNN architecture model shown in Figure 2; to classify visualized malware images. The TL-based VGG16 model is already trained on more than 14 million digital images of the ImageNet dataset. It is primarily introduced to resolve various identification challenges such as plant image, plankton, lung nodules classifications [42]. As observed in Figure 2, the DL VGG16 architecture comprises different connected CNN layers (16 layers) that are (1) five max-pooling (MP) layers, (2) five groups of convolutional (Conv.) layers, (3) three fully connected (FC) layers, and (4) a SoftMax output layer. Its input image size is 224 × 224 × 3, and its output layer includes the SoftMax classifier used for detection purposes.

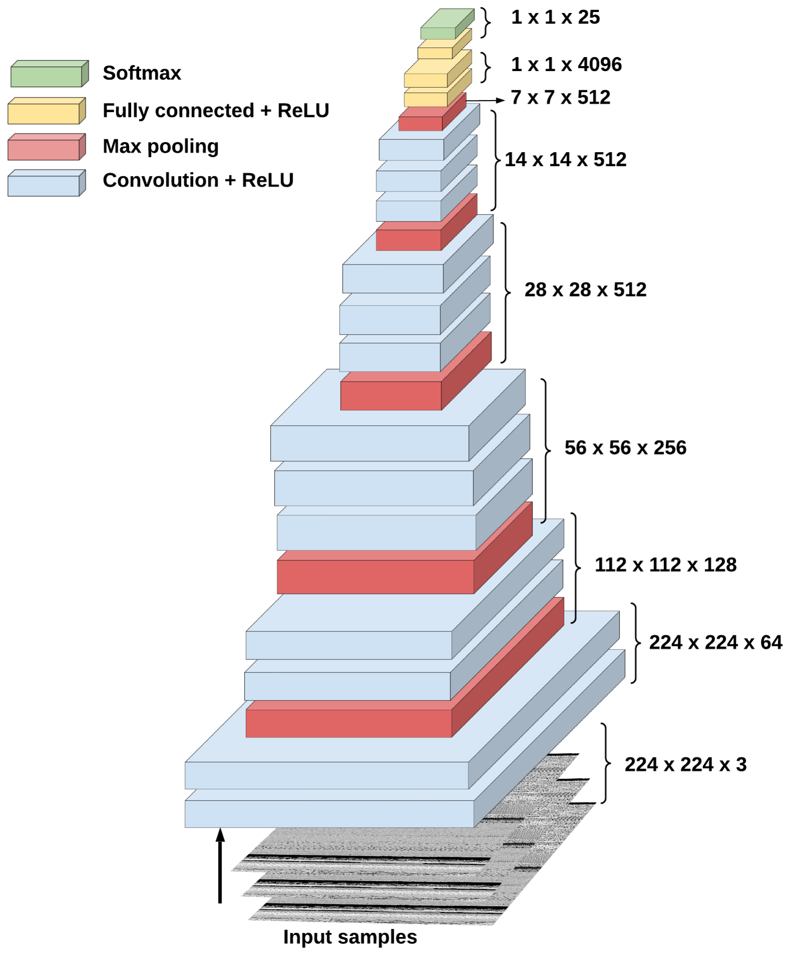


Figure 2. Architecture of the fine-tuned VGG16 model.

First, the malware images are resized to 224 × 224 × 3 to meet the input size of the first layer of the VGG16 model. Then, these images are passed to a group of convolutional layers with filter sizes of 3 × 3 and 1 × 1. In the convolution layers, the convolution stride is fixed to 1 padding for each 1 pixel. This is to ensure similar spatial dimensions among the included activation maps of the whole model layers. The rectified linear unit (ReLU) is used in all hidden layers to speed up the training process. The non-padding kernel filters of size 2 × 2 with two strides are applied in the max-pooling layers. For the output SoftMax layer, a classifier is used to classify the 25 malware families of the tested Malimg dataset. In the proposed model, all upper layers are frozen, and the last three connected layers are equipped to detect the malware family.

The prominent benefit of the VGG16 model is that it enhances the performance of CNNs without the necessity of doing deeper training with a high number of convolutional layers. This means that each convolutional layer will have various kernels that can learn and discover distinct image features with fewer iterations. Thus, it is computationally effective in malware image detection due to its low number of layers and iterations. More details about the architectures and explanations of the other seven pre-trained CNN models (AlexNet, DarkNet-53, DenseNet-201, Inception-V3, Places365-GoogleNet, ResNet-50, MobileNet-V2), could be explored in [43–49].

3.4. Fine-Tuning and Classification Phase

In general, TL can be performed in three different ways [50]: (1) shallow tuning: the last layer in the model is assigned to a new task, and the constraints of the other model's layers are frozen, (2) deep tuning: the whole end-to-end parameters of the pre-trained CNN architecture are retrained, and (3) fine-tuning: the CNN layers are gradually trained by fine-tuning the learning hyperparameters till a remarkable performance enhancement is accomplished. This paper applied fine-tuning, which is a compromised approach between the other two tuning types. Fine-tuning takes advantage of efficient classification tasks in the case of imbalanced datasets. Additionally, it causes lower complexity than the deep tuning and better classification accuracy than the shallow tuning.

The employed eight CNN models previously trained on the ImageNet dataset that contains 1000 different classes are adapted to our malware detection challenge. This adaptation is implemented through fine-tuning their layers' parameters and weights. In these models, the output layer that contains 1000 classes is modified and fine-tuned to comprise 25 classes (25 malware families). Additionally, as will be discussed, the original weights of the primary pre-trained CNN models of the ImageNet dataset were initially used, and after that, they were optimized and fine-tuned based on the back-propagation technique [51].

The fine-tuning process of the layers' weights is an iterative optimization procedure that is performed and repeated until determining the best value of the filter weights (w) that achieve a minimum error rate. The used cost function is expressed in Equation (1)

$$F(w, K) = \frac{1}{m}\sum_{j=1}^{m} f\left(p\left(k_j, w\right), \hat{c}_j\right) \tag{1}$$

where m refers to the malware images contained in the training dataset K, $p(k_j, w)$ is the CNN prediction function that predicts the class c_j of k_j by assuming the value of w, $\hat{c}_j$ is the proper class of the j^{th} malware image, k_j is the j^{th} malware image of K, and $f(c_j, \hat{c}_j)$ is the logistic error function that predicts c_j rather than $\hat{c}_i$.

During the fine-tuning process, the performance efficiency of three different optimizers [52]: (1) RMSprop (Root Mean Square Propagation), (2) SGDM (Stochastic Gradient Descent with Momentum), and (3) ADAM (Adaptive Moment Estimation Optimizer) were examined to select the best optimizer for estimating the optimal filter weights of the CNN layers. As a result, the ADAM optimizer was used for finding the optimal w due to its superior performance compared to other optimizers. ADAM optimizer combines the main advantages and benefits of the other two SGDM and RMSprop optimizers, where it establishes adaptive learning rates for each parameter in the training process. Hence, the significant improvement of the ADAM optimizer is that it retains an exponentially decaying average of the past squared gradient descent to reach a minimum value faster. Further details, mathematical expressions, and descriptions of the ADAM optimizer can be found in [53].

In the training process, a massive capacity of hardware memory is required to store the filter weights w of the CNN layers, so a mini-batch size is set to 64. The learning rate is regularly altered until the optimal value is reached; where high learning rates cause overfitting, while slow learning rates limit the error variants among epochs. Therefore, the initial learning rate was set to 0.00001; to efficiently regulate the update of the weight sizes. The CNN layers' weights were updated in each iteration, and the mini-batches were iterated for every epoch. The max-epoch was set to 20, where this value was selected through observing validation errors throughout fine-tuning process using various learning rates. Furthermore, the L2-regularization (ridge regression) technique [54] with weight decay (L2-regularizer) = 0.01; was adopted to enhance and optimize the performance of the CNN models and avoid overfitting problem while analyzing small training samples. Therefore, it achieves robust and faster classification of malware images. For all examined models, the FC and SoftMax classifiers were used to detect the 25 malware images.

3.5. Performance Evaluation Phase

In this phase, extensive performance analysis in terms of 15 different evaluation metrics is presented to assess the examined models. Thus, the performance of the eight fine-tuned CNN classifiers was evaluated through detection assessment metrics, including accuracy and loss curves, specificity (TNR) (true negative rate), confusion matrix, NPV (negative predictive value), AROC (Area under the receiver operating characteristic curve), FPR (false positive rate), validation accuracy, FNR (false negative rate), precision (PPV) (positive predictive value), FDR (false discovery rate), recall (TPR) (true positive rate), FOR (false omission rate), sensitivity, F1-Score, and misclassification rate. These classification performance metrics have been comprehensively used in the research community to offer exhaustive evaluations of classification approaches [55,56].

The mathematical expressions of these evaluation metrics are formulated as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

$$\text{Specificity (TNR)} = \frac{TN}{TN + FP} \tag{3}$$

$$\text{NPV} = \frac{TN}{TN + FN} \tag{4}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{5}$$

$$\text{FNR} = \frac{FN}{FN + TP} \tag{6}$$

$$\text{Precision (PPV)} = \frac{TP}{TP + FP} \tag{7}$$

$$\text{FDR} = \frac{FP}{FP + TP} \tag{8}$$

$$\text{Recall (TPR)} = \text{Sensitivity} = \frac{TP}{TP + FN} \tag{9}$$

$$\text{FOR} = \frac{FN}{FN + TN} \tag{10}$$

$$\text{F1-Score} = \frac{2TP}{2TP + FP + FN} \tag{11}$$

$$\text{Misclassification rate} = \frac{FP + FN}{TP + FP + TN + FN} \tag{12}$$

where *TP* (true positive) implies that both actual and predicted malware types are positive, *TN* (true negative) means that both actual and predicted malware types are negative, *FP* (false positive) implies that the actual malware type is negative, while the predicted malware type is positive, and *FN* (false negative) means that the actual malware type is positive, while the predicted malware type is negative.

The *TP*, *TN*, *FP*, and *FN* values can be estimated as shown in Figure 3, which is the confusion matrix of multi-classification tasks, which is different from the traditional confusion matrix of binary classification tasks. In this paper, we have a confusion matrix of classification with N malware families (classes). For example, considering a specific malware family F_x where both the actual and predicted values are given, the four different classification results that can be obtained are: true positive (green), true negative (yellow), false positive (blue), and false negative (red).

More details, debates, and explanations about these evaluation metrics can be explored in [57].

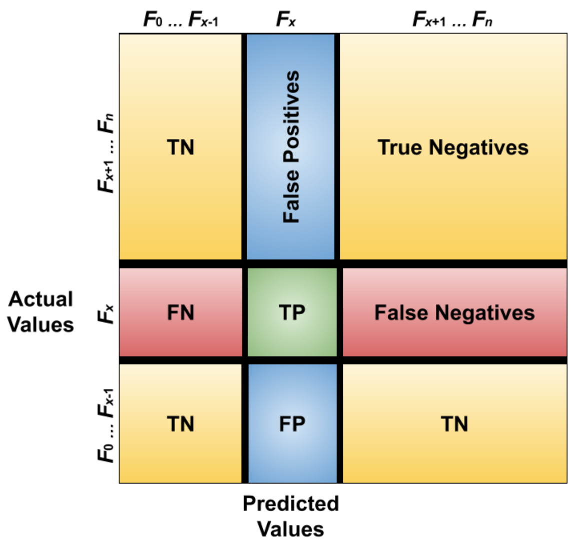


Figure 3. Multi-classification confusion matrix.

4. Experimental Results and Comparisons

This section presents and discusses the results of applying eight FT CNN-based TL models on an imbalanced malware dataset.

4.1. Malimg Dataset

The performance of the proposed FT CNN-based TL models was evaluated using the Malimg dataset [12]. Malimg consists of 9341 malware samples. Each sample corresponds to one of the 25 malware families as illustrated in Table 5. The malware samples of the Malimg dataset are visualized as gray-scale images within the range (0: black—255: white). Different fragments of the malware binary file demonstrate various image textures as shown in Figure 4. This figure illustrates the first fragment of the malware image which has a fine-grained texture since its corresponding .text section contains the malware executable code [6]. However, the remaining part of the .text section is visualized as a black block indicating zero paddings. Furthermore, the .data section, also, consists of a fine-grained texture that visualizes the initialized and uninitialized variables. Finally, the generated resources of the malware by the compiler are visualized in the .rsrc fragment. The Malimg dataset was deployed to train the CNN layers on the main characteristics of these families to identify and classify them correctly.

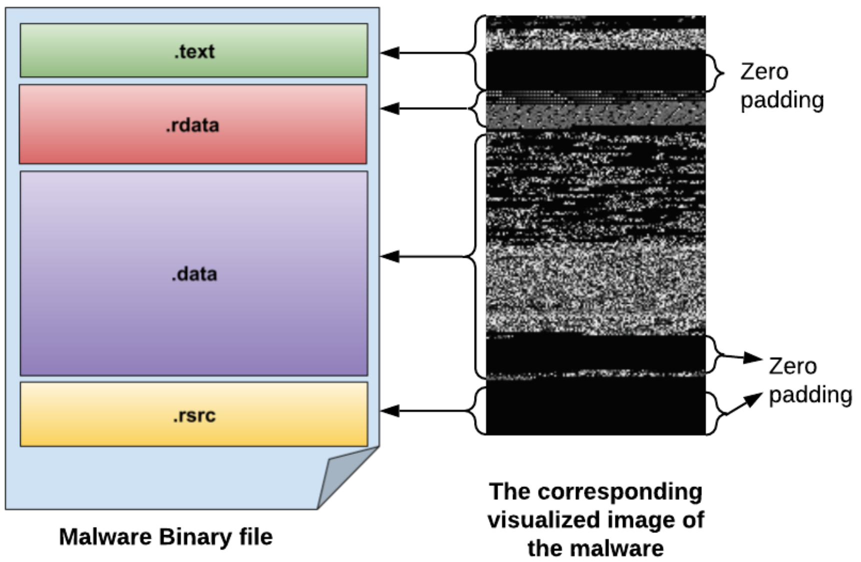


Figure 4. A sample of malware binary file sections and their corresponding fragments in the visualized version.

Table 5. The distribution of malware families among Malimg samples.

Malware Type	Family Class Name	Class ID	No. of Images	Percentage	Total
Worm	VB_AT	23	408	4.37%	5748
	Yuner_A	25	800	8.56%	
	Allaple_A	3	2949	31.57%	
	Allaple_L	4	1591	17.03%	
Trojan	Alueron_gen!J	5	198	2.12%	760
	C2LOP_gen!g	7	200	1.56%	
	C2LOP_P	8	146	2.14%	
	Malex_gen!J	17	136	1.46%	
	Skintrim_N	20	80	0.86%	
Dialer	Adialer_C	1	122	1.31%	730
	Dialplatform_B	9	177	1.89%	
	Instantaccess	12	431	4.61%	
PWS	Lolyda_AA1	13	213	2.28%	679
	Lolyda_AA2	14	184	1.97%	
	Lolyda_AA3	15	123	1.32%	
	Lolyda_AT	16	159	1.70%	
Trojan Downloader	Dontovo_A	10	162	1.73%	661
	Obfuscator_AD	18	142	1.52%	
	Swizzor_gen!E	21	128	1.37%	
	Swizzor_gen!I	22	132	1.41%	
	Wintrim_BX	24	97	1.04%	
Rogue	Fakerean	11	381	4.08%	381
Backdoor	Agent_FYI	2	116	1.24%	274
	Rbot!gen	19	158	1.69%	
Worm:AutoIT	Autorun_K	6	108	1.16%	108
				Total	9341

4.2. Results Analysis

In the simulation studies, we tested eight different CNN models (VGG16, AlexNet, DarkNet-53, DenseNet-201, Inception-V3, Places365-GoogleNet, ResNet-50, and MobileNet-V2) on the families of the Malimg dataset presented in Table 5. As discussed before, we divided the dataset as 80% for training and 20% for testing. As shown in Table 5, this dataset is imbalanced. Therefore, to handle this challenge, we applied the fine-tuning process for the CNN layers and hyperparameters without employing any data augmentation, as discussed in Section 3.4. The performance analysis of the simulation tests in terms of the confusion matrix, loss and accuracy curves, and other different evaluation metrics were carried out by MATLAB 2020b. An Intel Core i7-4500 processor with 8 GB RAM was used for training and classification processes.

For simplicity in presenting the simulation results, the accuracy and loss curves, the confusion matrix, and the other calculated evaluation metrics are presented in detail for VGG16, the best accomplished fine-tuned model among the eight examined CNN models. For the other fine-tuned CNN models, we give the average values of all tested evaluation metrics to provide deep comparisons among them.

The accuracy and loss curves of the training and testing processes of the fine-tuned VGG16 model across 20 epochs are shown in Figure 5. As can be observed from these curves, both loss and accuracy curves were steady before less than ten epochs. Additionally, there is no overfitting of the training samples because there is a complete similarity between the training and testing curves for the proposed fine-tuned VGG16 model. Therefore, by using a smaller number of epochs, the performance of the proposed fine-tuned VGG16 CNN model was superior. Likewise, we observed similar accuracy and loss curves for the other seven fine-tuned CNN models.

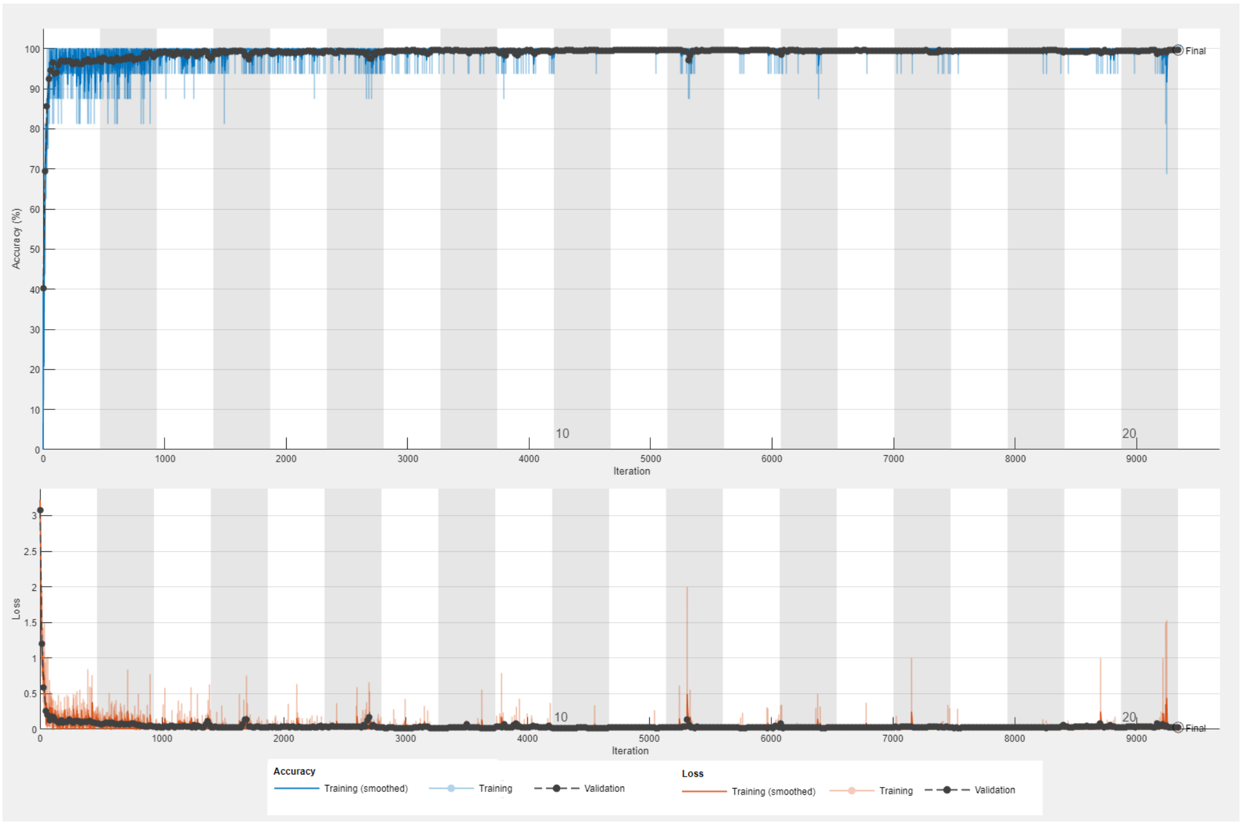
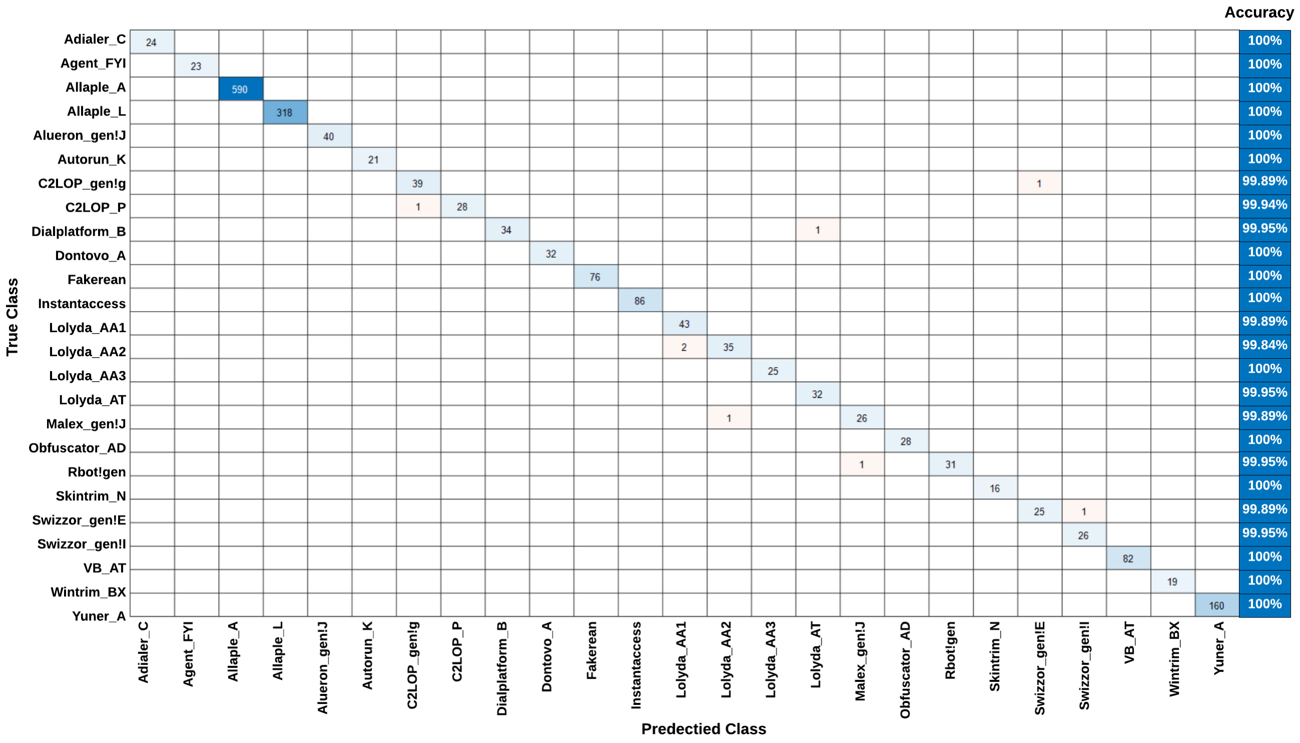


Figure 5. Accuracy and loss curves for the fine-tuned VGG16 classification model.

The confusion matrix obtained for the fine-tuned VGG16 model is shown in Figure 6. This is a multi-classification confusion matrix for the tested 25 malware families of the imbalanced Malimg dataset. The *TN*, *FN*, *TP*, and *FP* values for the individual detected malware families could be calculated with the aid of the multi-classification confusion matrix given in Figure 3. It is noticed that the attained *TN*, *FN*, *TP*, and *FP* values for the fine-tuned VGG16 model were as desired and close to the optimal values. This is due to the fine-tuning process of the CNN layers and hyperparameters. These achieved results confirm the low misclassification rate, where almost all malware families were correctly classified with high accuracy. Minimal families reported misclassification of their tested instances, which did not exceed two cases of malware images. The model succeeded in achieving 100% accuracy by 60% of the malware families where all their tested instances were classified correctly. For the rest of families most of them had only one misclassification such as 'C2LOP_gen!g' family with one *FN*; and only one family with 2 misclassifications which is 'Lolyda_AA2'.



	Adialer_C	Agent_FYI	Allaple_A	Allaple_L	Alueron_gen!J	Autorun_K	C2LOP_gen!g	C2LOP_P	Dialplatform_B	Dontovo_A	Fakerean	Instantaccess	Lolyda_AA1	Lolyda_AA2	Lolyda_AA3	Lolyda_AT	Malex_gen!J	Obfuscator_AD	Rbot!gen	Skintrim_N	Swizzor_gen!E	Swizzor_gen!I	VB_AT	Wintrim_BX	Yuner_A	Accuracy
Adialer_C	24																									100%
Agent_FYI		23																								100%
Allaple_A			590																							100%
Allaple_L				318																						100%
Alueron_gen!J					40																					100%
Autorun_K						21																				100%
C2LOP_gen!g							39														1					99.89%
C2LOP_P							1	28																		99.94%
Dialplatform_B									34							1										99.95%
Dontovo_A										32																100%
Fakerean											76															100%
Instantaccess												86														100%
Lolyda_AA1													43													99.89%
Lolyda_AA2													2	35												99.84%
Lolyda_AA3															25											100%
Lolyda_AT																32										99.95%
Malex_gen!J														1			26									99.89%
Obfuscator_AD																		28								100%
Rbot!gen																	1		31							99.95%
Skintrim_N																				16						100%
Swizzor_gen!E																					25	1				99.89%
Swizzor_gen!I																						26				99.95%
VB_AT																							82			100%
Wintrim_BX																								19		100%
Yuner_A																									160	100%

Figure 6. Confusion matrix of the fine-tuned VGG16 classification model.

We tested the capability of each fine-tuned CNN model of the eight examined models in classifying individual malware families. We calculated the accuracy (Acc.), precision (Prec.), recall (Rec.), specificity (Spec.), NPV, FPR, FNR, FDR, FOR, F1-Score, and misclassification rate (Mis. Class. Rate) for each malware family in the Malimg dataset. Then, we computed the average values of these evaluation metrics. Table 6 shows the obtained results of each malware family in addition to the overall average per evaluation metric after applying the fine-tuned VGG16 classifier. The results revealed that the proposed fine-tuned VGG16 model achieves significant values near optimum for all examined assessment metrics. Thus, this model is highly recommended to be applied in the context of efficient detection and classification of visualized malware samples.

Table 6. Evaluation metrics of the fine-tuned VGG16 classification model.

Family Name	Acc.	Prec. (PPV)	Rec. (TRP)	Spec. (TNR)	NPV	FPR	FNR	FDR	FOR	F1-Score	Mis Class. Rate
Adialer_C	1	1	1	1	1	0	0	0	0	1	0
Agent_FYI	1	1	1	1	1	0	0	0	0	1	0
Allaple_A	1	1	1	1	1	0	0	0	0	1	0
Allaple_L	1	1	1	1	1	0	0	0	0	1	0
Alueron_gen!J	1	1	1	1	1	0	0	0	0	1	0
Autorun_K	1	1	1	1	1	0	0	0	0	1	0
C2LOP_gen!g	0.9989	0.975	0.975	0.9994	0.9994	0.0005	0.025	0.025	0.0005	0.975	0.0011
C2LOP_P	0.9994	1	0.9655	1	0.9995	0	0.0345	0	0.0005	0.9825	0.0005
Dialplatform_B	0.9995	1	0.9714	1	0.9995	0	0.0286	0	0.0005	0.9855	0.0005
Dontovo_A	1	1	1	1	1	0	0	0	0	1	0
Fakerean	1	1	1	1	1	0	0	0	0	1	0
Instantaccess	1	1	1	1	1	0	0	0	0	1	0
Lolyda_AA1	0.9989	0.9556	1	0.9989	1	0.0011	0	0.0444	0	0.9773	0.0011
Lolyda_AA2	0.9984	0.9722	0.9459	0.9994	0.9989	0.0006	0.0541	0.0278	0.0011	0.9589	0.0016
Lolyda_AA3	1	1	1	1	1	0	0	0	0	1	0
Lolyda_AT	0.9995	0.9697	1	0.9995	1	0.0005	0	0.0303	0	0.9846	0.0005
Malex_gen!J	0.9989	0.9627	0.963	0.9995	0.9995	0.0005	0.037	0.037	0.0005	0.9629	0.0011
Obfuscator_AD	1	1	1	1	1	0	0	0	0	1	0
Rbot!gen	0.9995	1	0.9687	1	0.9995	0	0.0313	0	0.0005	0.9841	0.0005
Skintrim_N	1	1	1	1	1	0	0	0	0	1	0
Swizzor_gen!E	0.9989	0.9615	0.9615	0.9995	0.9995	0.0005	0.0385	0.0385	0.0005	0.9615	0.0011
Swizzor_gen!I	0.9995	0.963	1	0.9995	1	0.0005	0	0.037	0	0.9811	0.0005
VB_AT	1	1	1	1	1	0	0	0	0	1	0
Wintrim_BX	1	1	1	1	1	0	0	0	0	1	0
Yuner_A	1	1	1	1	1	0	0	0	0	1	0
Average	0.9997	0.9904	0.9901	0.9998	0.9998	0.0002	0.001	0.0096	0.0002	0.9902	0.0003

Moreover, Table 7 provides a comparison among the eight fine-tuned CNN models in terms of the evaluation metrics: Acc., Prec., Rec., Spec., NPV, FPR, FNR, FDR, FOR, F1-Score, and Mis. Class. Rate. In general, all employed CNN models achieved excellent classification results. Therefore, detecting malware images using imbalanced datasets can use these models effectively. Additionally, the results disclose again that the FT VGG16 model outperformed the other seven CNN models in almost all tested metrics.

Table 7. Average values of the evaluation metrics of the eight examined fine-tuned CNN classification models.

Evaluation Metric	VGG16	AlexNet	DarkNet-53	DenseNet-201	Inception-V3	Places365-GoogleNet	ResNet-50	MobileNet-V2
Acc.	0.9997	0.9996	0.9996	0.9996	0.9995	0.9995	0.9994	0.9994
Prec. (PPV)	0.9904	0.9857	0.9897	0.9868	0.9866	0.9834	0.9799	0.9775
Rec. (TPR)	0.9901	0.9858	0.986	0.9867	0.9838	0.9836	0.9789	0.9771
Spec. (TNR)	0.9998	0.9998	0.9998	0.9998	0.9998	0.9997	0.9997	0.9997
NPV	0.9998	0.9998	0.9998	0.9998	0.9998	0.9997	0.9997	0.9997
FPR	0.0002	0.0002	0.0002	0.0002	0.0002	0.0003	0.0003	0.0003
FNR	0.001	0.0142	0.014	0.0119	0.0162	0.0164	0.0211	0.0229
FDR	0.0096	0.0143	0.0103	0.0132	0.0134	0.0166	0.0201	0.0225
FOR	0.0002	0.0002	0.0002	0.0002	0.0002	0.0003	0.0003	0.0003
F1-Score	0.9902	0.9857	0.9877	0.9859	0.9832	0.9832	0.9785	0.977
Mis Class. Rate	0.0003	0.0004	0.0004	0.0004	0.0005	0.0005	0.0006	0.0006

Furthermore, Table 8 introduces a quantitative comparison of the computational overhead among the used CNN models in the proposed visualized malware multi-classification framework in terms of the (a) number of layers of the employed CNN model including the consecutive series or the parallel layers used for extracting features from the input malware images, (b) storage requirement which is defined as the allocated size of the employed model on the storage disk, (c) total number of the parameters in the original CNN model calculated based on the number of filters, weights, and stride values, (d) number of non-trainable parameters of the frozen layers in the used CNN-based TL model, (e) number of trainable parameters of the unfrozen layers in the employed CNN-based TL model, (f) total execution time of the training and validation processes of the employed CNN model (g), average time per malware sample which is calculated by dividing the total execution time by the total number of input malware samples, and finally (h) the percentage of reduced non-trainable parameters to the overall number of parameters of a specific CNN model. The obtained results in Table 8 show how the computational time of the training and validation processes various from one CNN model to another. The complexity variation is based on the used number of layers and the number of trainable parameters. In addition, the average time spent to classify the malware type and its family is adequate for all employed CNN models. This is due to the use of the transfer learning and the significant reduction in the training parameters that reaches 99.9% in some of the models. Most of the training parameters and layers were frozen, as discussed in Section 3.3. For example, the best accurate VGG16 CNN-based TL model used in the proposed framework achieved a low execution time of 0.7065 s to identify the malware sample. Only 102,400 parameters were trained from 138 million parameters that existed in the original CNN model.

Table 8. Experimental specifications and execution time of the tested FT CNN-based TL models.

CNN Model	No. of Layers	Storage Requirement (MB)	Total No. of Parameters (Millions)	Non-Trainable Parameters	Trainable Parameters	Total Execution Time (s)	Avg. Time per Malware (s)	Reduced Training Parameters (%)
VGG16	16	515	138	137,897,600	102,400	6600	0.7065	99.92
AlexNet	8	227	61	60,897,600	102,400	2340	0.2505	99.83
DarkNet53	53	155	41.6	41,574,400	25,600	4980	0.5331	99.93
DenseNet-201	201	167	20	19,952,000	48,000	8160	0.8736	99.76
InceptionV3	48	89	23.9	23,848,800	51,200	6420	0.6873	99.78
Places365-GoogleNet	22	27	7	6,974,400	25,600	2220	0.2377	99.63
ResNet50	50	96	25.6	25,548,800	51,200	3420	0.3661	99.80
MobileNetV2	53	13	3.5	3,468,000	32,000	3360	0.3597	99.08

4.3. Comparative Analysis

Many existing conventional ML and DL approaches use various data balancing methods to enhance the classification performance in the case of imbalanced dataset. In contrast, in this research, we used the imbalanced malware families of the benchmark Malimg database to classify malware images with high accuracy and without employing any data augmentation or any other balancing techniques.

Therefore, this section compares the performance of the proposed multi-classification framework with numerous well-established ML and DL classifiers; to prove its efficiency in detecting and classifying visualized malware images even when applied to same datasets with imbalanced malware families.

Table 9 summarizes the performance comparison in terms of the accuracy, precision, recall, and F1-Score metrics of the proposed multi-classification framework using fine-tuned VGG16 CNN model and the baseline-related ML and DL classifiers. The comparison results confirm that the developed multi-classification framework outperforms all other conventional approaches in all tested evaluation metrics. The classification accuracy of the proposed framework reached 99.97% after applying fine-tuned VGG16 which is considered superior compared to traditional classifiers. This is due to the use of an efficient FT process and well-developed CNN-based TL models.

Table 9. Comparative analysis between the proposed multi-classification framework and related ML and DL-based malware multi-classification approaches tested on the Malimg dataset.

Model	Acc. (%)	Prec. (%)	Rec. (%)	F1-Score (%)
[5]	97.55	97.43	97.50	97.46
[6]	98.65	98.86	98.63	98.74
[7]	99.14	-	-	-
[8]	99.30	-	96.00	-
[9]	97.68	98.00	98.00	98.00
[29]	98.40	-	-	97.45
[31]	98.82	98.85	98.81	98.75
[32]	99.67	99.02	-	-
[33]	98.79	98.79	98.47	98.46
[34]	98.70	-	-	-
[35]	98.11	98.53	98.64	98.24
[38]	98.91	99.00	99.00	99.00
[39]	97.00	97.00	97.00	97.00
[40]	98.00	98.20	98.40	97.10
Proposed	99.97	99.04	99.01	99.02

5. Conclusions and Future Work

There is an ongoing competition between anti-malware software and cyber-attackers' methods. Malware is one of the most widespread cyber-attacks on the Internet. Consequently, it is essential to continue building innovative, intelligent security tools to mitigate these security attacks. Thus, efficient artificial intelligence (AI) tools are designed and used to detect malicious software. Unfortunately, AI-based anti-malware solutions based on ML algorithms introduce considerable development costs by generating an extensive set of handcrafted features identification and extraction, which requires the time and expertise of data scientists and malware analysts.

On the other hand, AI-based anti-malware solutions based on DL algorithms and CNN architectures have exhibited outstanding performance in identifying malware quickly and efficiently. Therefore, this paper introduced a DL-based visualized malware multi-classification framework to classify different unbalanced families of malware images. This framework was built based on malware visualization, fine-tuning, and CNN-based transfer learning phases that were well-developed to accurately detect different categories of malware families.

The proposed framework comprises eight fine-tuned CNN models, VGG16, AlexNet, DarkNet-53, DenseNet-201, Inception-V3, Places365-GoogleNet, ResNet-50, and MobileNet-V2, that were already pre-trained on the ImageNet database. The main contribution of the proposed framework is the cost-effectiveness in handling the imbalanced malware families while achieving high detection performance and without the need for data augmentation processes or complex features engineering. Extensive simulation experiments based on various evaluation metrics were conducted on the benchmark imbalanced Malimg dataset, which proved the outstanding classification capability and proficiency of the proposed framework.

Furthermore, a comprehensive comparative analysis among the proposed work and recent well-known ML and DL-based malware classification algorithms was presented and discussed. The comparison results demonstrated that the proposed framework achieved superior outcomes for all examined classification metrics.

For future work, different balanced and imbalanced malware datasets can be tested and explored. Additionally, building and testing a new malware dataset with the recent well-known malicious software is one of our aims to be considered. Moreover, we intend to investigate and examine the detection of cyber-attacks and malicious software in IoT cybersecurity applications.

Author Contributions: Conceptualization, W.E.-S. and I.A.; methodology, W.E.-S. and I.A.; software, W.E.-S. and A.A.; validation, W.E.-S., I.A. and A.A.; formal analysis, W.E.-S. and I.A.; investigation, W.E.-S., I.A. and A.A.; resources, W.E.-S., I.A. and A.A.; data curation, W.E.-S., I.A. and A.A.; writing—original draft preparation, W.E.-S. and A.A.; writing—I.A.; visualization, W.E.-S., I.A. and A.A.; supervision, W.E.-S. and I.A.; project administration, W.E.-S. and I.A.; funding acquisition, I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Prince Sultan University, Saudi Arabia, [Grant number: SEED-CCIS-2021-84].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication.

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

References

1. Komatwar, R.; Kokare, M. A survey on malware detection and classification. *J. Appl. Secur. Res.* **2020**, 1–31. [CrossRef]
2. McAfee Report Threat Center. Available online: https://www.mcafee.com/enterprise/en-us/threat-center/mcafee-labs/reports.html (accessed on 3 June 2021).
3. Almomani, I.; Khayer, A. Android applications scanning: The guide. In Proceedings of the 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 3–4 April 2019; pp. 1–5.
4. Almomani, I.M.; Al Khayer, A. A Comprehensive Analysis of the Android Permissions System. *IEEE Access* **2020**, *8*, 216671–216688. [CrossRef]
5. 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]
6. Roseline, S.A.; Geetha, S.; Kadry, S.; Nam, Y. Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm. *IEEE Access* **2020**, *8*, 206303–206324. [CrossRef]

7. Moussas, V.; Andreatos, A. Malware Detection Based on Code Visualization and Two-Level Classification. *Information* **2021**, *12*, 118. [CrossRef]
8. 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]
9. Roseline, S.A.; Hari, G.; Geetha, S.; Krishnamurthy, R. Vision-Based Malware Detection and Classification Using Lightweight Deep Learning Paradigm. In Proceedings of the International Conference on Computer Vision and Image Processing, Jaipur, India, 27–29 September 2019; Springer: Singapore, 2019; pp. 62–73.
10. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
11. Basha, S.S.; Vinakota, S.K.; Pulabaigari, V.; Mukherjee, S.; Dubey, S.R. Autotune: Automatically tuning convolutional neural networks for improved transfer learning. *Neural Netw.* **2021**, *133*, 112–122. [CrossRef]
12. 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.
13. Al Khayer, A.; Almomani, I.; Elkawlak, K. ASAF: Android Static Analysis Framework. In Proceedings of the 2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 3–5 November 2020; pp. 197–202.
14. Nassiri, M.; HaddadPajouh, H.; Dehghantanha, A.; Karimipour, H.; Parizi, R.M.; Srivastava, G. Malware elimination impact on dynamic analysis: An experimental machine learning approach. In *Handbook of Big Data Privacy*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 359–370.
15. Martins, N.; Cruz, J.M.; Cruz, T.; Abreu, P.H. Adversarial machine learning applied to intrusion and malware scenarios: A systematic review. *IEEE Access* **2020**, *8*, 35403–35419. [CrossRef]
16. Qaddoura, R.; Aljarah, I.; Faris, H.; Almomani, I. A Classification Approach Based on Evolutionary Clustering and Its Application for Ransomware Detection. In *Evolutionary Data Clustering: Algorithms and Applications*; Springer: Singapore, 2021; pp. 237–248. [CrossRef]
17. Yang, Y.; Du, X.; Yang, Z.; Liu, X. Android Malware Detection Based on Structural Features of the Function Call Graph. *Electronics* **2021**, *10*, 186. [CrossRef]
18. Poudyal, S.; Gupta, K.D.; Sen, S. PEFile analysis: A static approach to ransomware analysis. *Int. J. Comput. Sci.* **2019**, *1*, 34–39.
19. Jose, R.R.; Salim, A. Integrated Static Analysis for Malware Variants Detection. In Proceedings of the International Conference on Inventive Computation Technologies, Coimbatore, India, 29–30 August 2019; Springer: Cham, Switzerland, 2019; pp. 622–629.
20. Nar, M.; Kakisim, A.G.; Yavuz, M.N.; Soğukpinar, İ. Analysis and comparison of disassemblers for opcode based malware analysis. In Proceedings of the 2019 4th International Conference on Computer Science and Engineering (UBMK), Samsun, Turkey, 11–15 September 2019; pp. 17–22.
21. Alsoghyer, S.; Almomani, I. Ransomware detection system for Android applications. *Electronics* **2019**, *8*, 868. [CrossRef]
22. Alsoghyer, S.; Almomani, I. On the Effectiveness of Application Permissions for Android Ransomware Detection. In Proceedings of the 2020 6th Conference on Data Science and Machine Learning Applications (CDMA), Riyadh, Saudi Arabia, 4–5 March 2020; pp. 94–99. [CrossRef]
23. Faris, H.; Habib, M.; Almomani, I.; Eshtay, M.; Aljarah, I. Optimizing extreme learning machines using chains of salps for efficient Android ransomware detection. *Appl. Sci.* **2020**, *10*, 3706. [CrossRef]
24. Almomani, I.; AlKhayer, A.; Ahmed, M. An Efficient Machine Learning-based Approach for Android v. 11 Ransomware Detection. In Proceedings of the 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), Riyadh, Saudi Arabia, 6–7 April 2021; pp. 240–244.
25. Jeon, J.; Kim, J.; Jeon, S.; Lee, S.; Jeong, Y.S. Static Analysis for Malware Detection with Tensorflow and GPU. In *Advances in Computer Science and Ubiquitous Computing*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 537–546.
26. Amer, E.; Zelinka, I. A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence. *Comput. Secur.* **2020**, *92*, 101760. [CrossRef]
27. Mohaisen, A.; Alrawi, O.; Mohaisen, M. AMAL: High-fidelity, behavior-based automated malware analysis and classification. *Comput. Secur.* **2015**, *52*, 251–266. [CrossRef]
28. Sihwail, R.; Omar, K.; Zainol Ariffin, K.A.; Al Afghani, S. Malware detection approach based on artifacts in memory image and dynamic analysis. *Appl. Sci.* **2019**, *9*, 3680. [CrossRef]
29. 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]
30. Jang, S.; Li, S.; Sung, Y. Fasttext-based local feature visualization algorithm for merged image-based malware classification framework for cyber security and cyber defense. *Mathematics* **2020**, *8*, 460. [CrossRef]
31. 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]
32. 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]

33. 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]
34. Saadat, S.; Raymond, V.J. Malware Classification Using CNN-XGBoost Model. In *Artificial Intelligence Techniques for Advanced Computing Applications*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 191–202.
35. 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]
36. Narayanan, B.N.; Davuluru, V.S.P. Ensemble malware classification system using deep neural networks. *Electronics* **2020**, *9*, 721. [CrossRef]
37. Almomani, I.; Qaddoura, R.; Habib, M.; Alsoghyer, S.; Khayer, A.A.; Aljarah, I.; Faris, H. Android ransomware detection based on a hybrid evolutionary approach in the context of highly imbalanced data. *IEEE Access* **2021**. [CrossRef]
38. Roseline, S.A.; Sasisri, A.; 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), Chennai, India, 1–3 October 2019; pp. 1–6.
39. Ouahab, I.B.A.; Bouhorma, M.; Boudhir, A.A.; El Aachak, L. Classification of Grayscale Malware Images Using the K-Nearest Neighbor Algorithm. In *Proceedings of the Third International Conference on Smart City Applications*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 1038–1050.
40. 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]
41. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
42. Theckedath, D.; Sedamkar, R. Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks. *Sn Comput. Sci.* **2020**, *1*, 1–7. [CrossRef]
43. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
44. Hong, F.; Lu, C.; Jiang, W.; Ju, W.; Wang, T. RDNet: Regression Dense and Attention for Object Detection in Traffic Symbols. *IEEE Sens. J.* **2021**. [CrossRef]
45. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
46. 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.
47. Zhou, B.; Lapedriza, A.; Khosla, A.; Oliva, A.; Torralba, A. Places: A 10 million image database for scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 1452–1464. [CrossRef]
48. Rezende, E.; Ruppert, G.; Carvalho, T.; Ramos, F.; De Geus, P. Malicious software classification using transfer learning of resnet-50 deep neural network. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 1011–1014.
49. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
50. El-Hag, N.A.; Sedik, A.; El-Shafai, W.; El-Hoseny, H.M.; Khalaf, A.A.; El-Fishawy, A.S.; Al-Nuaimy, W.; Abd El-Samie, F.E.; El-Banby, G.M. Classification of retinal images based on convolutional neural network. *Microsc. Res. Tech.* **2021**, *84*, 394–414. [CrossRef] [PubMed]
51. Hegazy, T.; Fazio, P.; Moselhi, O. Developing practical neural network applications using back-propagation. *Comput. Aided Civ. Infrastruct. Eng.* **1994**, *9*, 145–159. [CrossRef]
52. Bera, S.; Shrivastava, V.K. Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. *Int. J. Remote Sens.* **2020**, *41*, 2664–2683. [CrossRef]
53. Jais, I.K.M.; Ismail, A.R.; Nisa, S.Q. Adam optimization algorithm for wide and deep neural network. *Knowl. Eng. Data Sci.* **2019**, *2*, 41–46. [CrossRef]
54. Gao, H.; Yang, Y.; Lei, S.; Li, C.; Zhou, H.; Qu, X. Multi-branch fusion network for hyperspectral image classification. *Knowl. Based Syst.* **2019**, *167*, 11–25. [CrossRef]
55. Ni, S.; Qian, Q.; Zhang, R. Malware identification using visualization images and deep learning. *Comput. Secur.* **2018**, *77*, 871–885. [CrossRef]
56. Namanya, A.P.; Awan, I.U.; Disso, J.P.; Younas, M. Similarity hash based scoring of portable executable files for efficient malware detection in IoT. *Future Gener. Comput. Syst.* **2020**, *110*, 824–832. [CrossRef]
57. Stamp, M.; Alazab, M.; Shalaginov, A. *Malware Analysis Using Artificial Intelligence and Deep Learning*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 1–655.