applied sciences

MDPI

Article

Examining the Performance of Various Pretrained Convolutional Neural Network Models in Malware Detection

Falah Amer Abdulazeez [1], Ismail Taha Ahmed [2,*] and Baraa Tareq Hammad [2]

[1] College of Education Pure Sciences, University of Anbar, Anbar 55431, Iraq; falah.amer.azeez@uoanbar.edu.iq
[2] College of Computer Sciences and Information Technology, University of Anbar, Anbar 55431, Iraq; baraa.tareq@uoanbar.edu.iq
* Correspondence: ismail.taha@uoanbar.edu.iq

Abstract: A significant quantity of malware is created on purpose every day. Users of smartphones and computer networks now mostly worry about malware. These days, malware detection is a major concern in the cybersecurity area. Several factors can impact malware detection performance, such as inappropriate features and classifiers, extensive domain knowledge, imbalanced data environments, computational complexity, and resource usage. A significant number of existing malware detection methods have been impacted by these factors. Therefore, in this paper, we will first identify and determine the best features and classifiers and then use them in order to propose the malware detection method. The comparative strategy and proposed malware detection procedure consist of four basic steps: malware transformation (converting images of malware from RGB to grayscale), feature extraction (using the ResNet-50, DenseNet-201, GoogLeNet, AlexNet, and SqueezeNet models), feature selection (using PCA method), classification (including GDA, KNN, logistic, SVM, RF, and ensemble learning), and evaluation (using accuracy and error evaluation metrics). Unbalanced Malimg datasets are used in experiments to validate the efficacy of the results that were obtained. According to the comparison findings, KNN is the best machine learning classifier. It outperformed the other classifiers in the Malimg datasets in terms of both accuracy and error. In addition, DenseNet201 is the best pretrained model in the Malimg dataset. Therefore, the proposed DenseNet201-KNN methods had an accuracy rate of 96% and a minimal error rate of 3.07%. The proposed methods surpass existing state-of-the-art approaches. The proposed feature extraction is computationally quicker than most other methods since it uses a lightweight design and fewer feature vector dimensions.

Keywords: malware detection; pretrained CNN models; unbalanced malimg datasets; DenseNet201-KNN

Citation: Abdulazeez, F.A.; Ahmed, I.T.; Hammad, B.T. Examining the Performance of Various Pretrained Convolutional Neural Network Models in Malware Detection. *Appl. Sci.* **2024**, *14*, 2614. https://doi.org/10.3390/app14062614

Academic Editor: Luis Javier Garcia Villalba

Received: 21 February 2024
Revised: 15 March 2024
Accepted: 18 March 2024
Published: 20 March 2024

1. Introduction

Malware, often known as malicious software, is software that is purposefully created to damage a computer system. Malware is typically divided into several categories, as shown in Figure 1. It is employed in offenses to penetrate or gain access to certain digital assets that are highly sensitive or may create harm or undesired consequences to users' systems. Malware attacks often target traditional internet-connected computers, but they can also target IoT and smart devices. Therefore, intelligent cybersecurity techniques are needed to protect millions of IoT users from malicious attacks. Malware analysis is the study of the unique characteristics, targets, sources, and potential impact of malicious software and code, such as spyware, viruses, malvertising, and ransomware. Malware code is analyzed to understand how it differs from other types [1,2].

The most common technique to classify malware uses static and dynamic analysis. Figure 2 shows a taxonomy for malware analysis that is frequently used. Static analysis captures information without running the malware binary. When performing dynamic analysis, it is necessary to pay attention to the activity of malware while it is running.

Despite the fact that dynamic analysis is generally seen as more dependable and effective over a longer period of time, it has significant restrictions. For example, malware analysis that allows the malware to serve a purpose is time-consuming and cannot be deployed to endpoints in real time because results take too long [3]. Malware detection and classification are two categories of malware analysis. Malware can be classified as either malicious or benign via the use of detection. Malware classification involves identifying the particular family of malware that is being discussed.

Figure 1. Malware types [1].

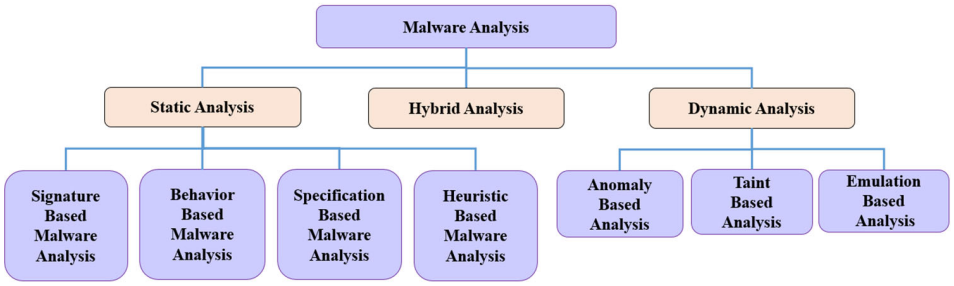


Figure 2. The most current categories of malware analysis [4].

Most malware detection methods available today in terms of features may be separated into two distinct groups: the hand-crafted-feature-based group and the deep-crafted-feature-based group. The feature extraction procedure in hand-crafted methods includes manually choosing the feature kind. In contrast, malware may be automatically identified and classified using deep learning techniques. One of the features of traditional malware detection methods is that they are often hand-crafted. Unfortunately, a lot of them depend on extensive domain knowledge, which is frequently time-consuming. This presents a challenge because the growth rate of new malware is so fast that this method cannot keep up with the rate at which malware is being created. Nevertheless, the majority of widely recognized anti-malware products employed the aforementioned methods, usually in conjunction with signature-based techniques that create local databases that store malware signature patterns.

Machine learning (ML) models are used in many fields these days. Additionally, modern platforms utilize image processing and machine learning or deep learning-based approaches to classify malware. The use of machine learning techniques and artificial neural networks (ANNs) is a typical option. These methods can be adapted to more complicated designs, such as the use of ensemble learning to recognize malware based on information collected from its properties.

By allowing representation learning to use raw input data, deep learning techniques facilitate automatic learning. Deep learning models detect complex patterns in images, text, audio, and other data and provide accurate insights and predictions. Figure 3 shows the classification of deep learning architecture. For the purpose of addressing demanding

feature engineering challenges and the requirement for domain expertise, deep learning and image processing approaches have been utilized for the classification of malware.

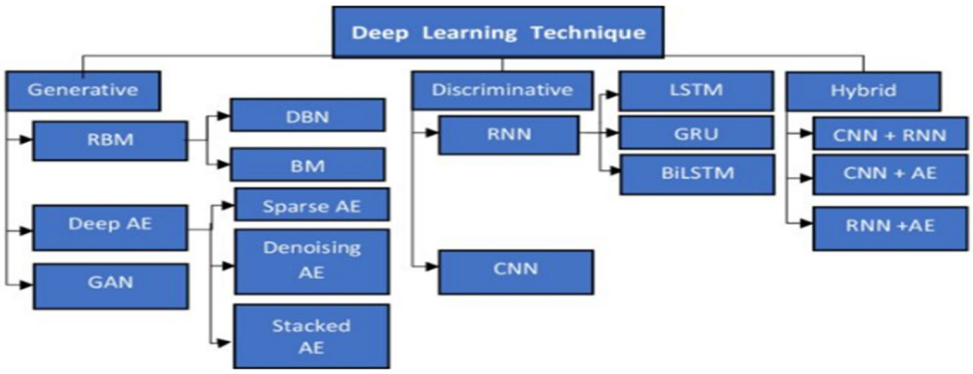


Figure 3. Classification of deep learning architecture [5].

Convolutional neural networks (CNNs) are one of the most popular deep learning methods. They are used for image classification. One can overlook manual feature development because CNN provides a more accurate depiction of the data. Because of its multilayer construction, CNN deep learning is incredibly efficient in learning the features in both labeled and unlabeled input data. Unfortunately, utilizing the CNN deep learning model requires us to constantly train the model on a sizable data set, which is time-consuming and computationally demanding. In order to curb this learning challenge, the knowledge gained via CNNs can be converted to a variety of learning tasks to enhance malware detection and classification. The primary goal of transfer learning is to it enable task training with a small dataset via the use of a model previously trained on a large set of data. Consequently, we can extract features using CNN architectures that have already been trained before creating the classification model. This encourages us to generate a pretrained CNN model for the purpose of identifying and classifying malware classes. Numerous pretrained CNN models play a significant role in many different applications. These models include MobileNetV2, ResNet-50, VGG16, DenseNet-201, ResNet50, SqueezeNet, InceptionV3, Xception, VGG19, DenseNet, and AlexNet.

Pretrained model features are used by most malware detection methods. Although most malware detection concerns have been resolved by earlier approaches, the current approaches frequently have three key problems: (1) extracting dependable and effective features that are effective in detecting malware; (2) large weight; and (3) database imbalances that impede the proper identification and classification of malware. The most difficult task is developing a malware detection model that relies on the aforementioned issues and is effective. Thus, the purpose of this work is to identify and determine the best pretrained model features and classifiers used in malware detection. The proposed malware detection approach can make use of these specific features and classifiers. The main goals of the recommended method are to improve malware detection accuracy, decrease the likelihood of inaccurate classification, and extract relevant information.

The following are the contributions of the proposed method:

1. The result of the comparison is to identify the best pretrained CNN model features and classifiers to propose a malware detection technique.
2. Collect and extract the most valuable features from the last fully connected layers of ResNet50, DenseNet201, GoogLeNet, AlexNet, and SqueezeNet models.
3. Simplifying the training procedure by employing models that have already been trained as feature extractors, such as AlexNet, ResNet50, GoogLeNet, SqueezeNet, and DenseNet201, and then using PCA to generate finer features.
4. According to all evaluation metrics, the proposed malware detection DenseNet201-KNN yielded the most successful results using the Malimg dataset.

5. The lightweight design used by each of the proposed malware detection techniques requires a little time and resources.

The following are the rest of the sections of the paper: Section 2 contains a discussion of the works that are related. A description of the features of the pretrained CNN models is given in Section 3. Section 4 presents the details of the proposed method. The experimental findings, their interpretation, and the significance of them are given in Section 5. Section 6 concludes the paper.

2. Related Works

Here, feature engineering is not necessary for the following methods. We employed image unprocessed pixel information as our foundational malware representation. Furthermore, to improve the classification and detection outcomes, we used knowledge transfer from a deep neural network trained for object detection tasks on a different dataset to find good malware descriptions. Numerous academics have looked into the use of machine learning techniques and deep learning models for malware detection. Convolutional neural networks (CNNs) are one of the most widely used deep learning techniques because they make feature extraction simpler to perform. Researchers used deep learning as a method for feature extraction for identifying malware in order to improve the classifier's performance. CNN structures that were previously trained before building the classification model can be used to extract the features. The role of the pretrained CNN model is to recognize and categorize different types of malware.

In this section, a full study of several of the most important malware detection works relying on pretrained CNN model features was analyzed. Rezende et al. [6] proposed a malware detection method based on a pretrained VGG16 model. Using the ImageNet dataset as training data, they first retrieved features from the bottleneck layer of VGG16. During the classification stage, the SVM classifier was employed. Vasan et al. [7] proposed a malware detection method based on a pretrained ResNet50 model. A PCA procedure was utilized in order to decrease the dimension of the feature vector. Several classification techniques, including CNN architecture ensemble, SoftMax, and multiclass SVMs, were employed throughout the classification step. Pant et al. [8] proposed a malware detection method based on a pretrained VGG16 model. The proposed method was tested using the Malimg Database. Customized CNN outperformed VGG16, ResNet-18, and InceptionV3 in terms of accuracy because of its capacity to process small and non-uniform input for the classification of malware. Kumar et al. [9] Proposed malware detection based on pretrained CNN models that had already been trained on ImageNet for classification. An evaluation of the proposed method was carried out using the Malimg and BIG2015 databases. Compared to pretrained models like VGG16, VGG19, ResNet50, and InceptionV3, the customized CNN model performed better. Gyamfi et al. [10] presented a comparative analysis of several pretrained models, including the VGG16, VGG19, DenseNet, and AlexNet models. Principal component analysis (PCA) is used to apply feature extraction. Next was the Particle Swarm Optimization (PSO) feature selection process. The proposed method was tested using the Malimg database. A CNN was used in the classification stage. Asam et al. [11] presented a malware detection method relying on ResNet-18 and DenseNet201 models. The second layer of the classification layer's deep features were put together to create a boosted feature set, which was then given to SVM. Aslan et al. [12] presented a malware detection method relying on merging two pretrained models, such as ResNet and Alex Net. The proposed method was tested using the Malimg database. Despite achieving a high accuracy percentage, the generated vector had large dimensions (4096). Hammad et al. [13] proposed a malware detection method based on the GoogLeNet model. Several classification algorithms, such as KNN, SVM, and ELM, were employed during the classification stage. The proposed method was examined on common unbalanced Malimg datasets. The accuracy percentage of the presented method exceeded the previous methods, namely that of the deep-feature and hand-crafted feature methods. Lastly, Table 1 provides a summary of the current appropriate works in the malware detection field.

Table 1. Summary of previous studies of malware detection approaches (2018–2023).

Ref	Year	Model as Feature Extractor	Classifier	DB	Limitations
Khan et al. [14]	2018	GoogLeNet, ResNet18	CNN	EXE files as image	Significant weight, deep understanding, longer execution times, and significant validation losses
Rezende et al. [6]	2018	VGG16	SVM	Malimg	High feature vector dimension
Lo et al. [15]	2019	Xception	Ensemble model	Malimg, Microsoft Malware	Significant weight, deep understanding, longer execution times, and significant validation losses
Singh et al. [16]	2019	ResNet-50	CNN	collected dataset, Malimg	Obfuscation prevents visibility, low for packed or unnoticed, and undetectable evasive malware
Vasan et al. [17]	2020	VGG16, ResNet-50, Inception	CNN	Malimg, IoT-Android Mobile	Requires comprehensive expertise in the field and high dimensions of the feature vectors
Aslan et al. [12]	2021	ResNet and AlexNet	KNN, SVM, RF, NB	Malimg, Microsoft BIG 2015, and MaleVis	Despite achieving a high accuracy percentage, the generated vector has large dimensions (4096)
Marastoni et al. [18]	2021	CNN	LSTM	OBF, Malimg, and MsM2015	Fixed dimensions and obfuscation methods
Anandhi et al. [19]	2021	DenseNet201, VGG3	Densely Connected Network	Malimg, BIG 2015	Significant weight and consistent image size
Pant et al. [8]	2021	Custom CNN, VGG16, Resnet-18, Inception-V3	CNN	Malimg	Poor pretrained model, inconsistent data, and not enough information
Kumar et al. [9]	2022	VGG16, VGG19, ResNet50, Inception V3	CNN	Malimg, Microsoft BIG	Significant weight and challenging fine-tuning
Asam et al. [20]	2022	AlexNet, VGG16, ResNet50, Xception, GoogLeNet	SoftMax	IoT Dataset	CNN architecture is complicated and time-consuming
Shaukat et al. [21]	2023	15 models	12 Classifiers	Malimg	Despite achieving a high accuracy percentage, the generated vector has large dimensions
Dawra et al. [22]	2023	ResNet50, VGG19, Xception	CNN	Malimg	High feature vector dimension

3. Pretrained CNN Models Overview

The motivation to utilize deep learning in the majority of studies is due to the importance of deep learning methods in the areas of computer vision and GPU development. In DL, the phases of feature extraction and classification are combined. The aforementioned techniques have been demonstrated to be much more precise than traditional methods because they enable the automatic learning of complex and abstract features. Additionally, it cuts down on the time and effort required to locate meticulously crafted details in malware images. Examples of deep learning models are convolutional neural networks (CNNs), deep neural networks (DNNs), and recurrent neural networks (RNNs). CNN is one of the most often used DL models. While there are other deep learning techniques, the most widely used ones are recurrent neural networks (RNNs) and convolutional neural networks (CNNs). CNNs and RNNs are specialized types of deep neural networks (DNNs). There are distinctions in the architectures of the neural networks; CNNs are more adapted to analyze spatial data, like photos, while RNNs have been more appropriate for analyzing temporal and sequential data, like text or videos. This explains why CNNs are currently surpassing RNNs.

A CNN's convolution layer performs dual functions of feature extraction and discrimination [23]. Deep learning model training is challenging and requires a large amount of knowledge and processing power. In order to minimize the training cost, pretrained CNN models were implemented as feature extractors. Several pretrained CNN models were developed, including DenseNet-201, GoogLeNet, ResNet-50, AlexNet, and SqueezeNet. We choose these models based on varying feature dimensions, years, layers, and parameters. These CNN models that were previously trained have been deployed and utilized for various other multiclass classification tasks. The CNN models are trained on vast datasets, such as the ImageNet database, which comprises over a million labeled high-resolution images across approximately 22,000 categories. We shall provide a brief description of the pretrained models employed in this paper in the following subsections. Table 2 summarizes the essential characteristics of several pretrained networks.

Table 2. The most common pretrained CNN model properties.

Model	ResNet50	GoogLeNet	AlexNet	DenseNet201	SqueezeNet
Year	2016	2014	2012	2016	2016
Image Dimensions	224 × 224 × 3	224 × 224 × 3	227 × 227 × 3	224 × 224 × 3	227 × 227
Number of layers	177	22	8	201	18
Number of Parameters	23 million	4 million	60 million	20 million	1.25 million
Feature Dimension	2048	5643	4096	1000	

3.1. DenseNet201 Model as Feature Extractor

Huang et al. [24] introduced a novel concept known as Dense Block, which adds feature reuse across the network [25]. This is the basis for DenseNet-201's design. Due to the fact that it contains 201 layers overall, it is called "DenseNet-201". Every layer in a Dense Net architecture receives inputs from every layer that preceded it, and it then sends its outputs to every layer that succeeds it. Each layer will always have immediate access to the gradient signals and characteristics that all of the layers before it learned, owing to this connectivity pattern [26]. Figure 4 depicts the DenseNet201 architecture.

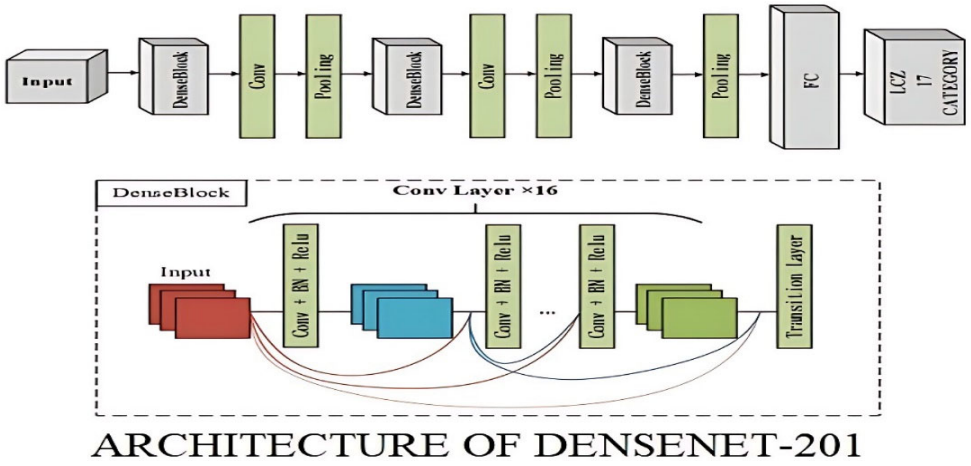


Figure 4. The fundamental design of DenseNet201 [24].

3.2. ResNet50 Model as Feature Extractor

ResNet (Residual Network) is a shortened variant of residual networks with 50 layers [27]. He and colleagues [28] created ResNet [28]. Different kinds of ResNet topologies were created, with the variety of layers used ranging from 34 to higher levels. Among the remaining models, ResNet50—which has 49 convolutional layers and one fully connected (FC) layer—became extensively used. While there were 3.9 million MACs (multiply accumulate operations) overall, there were 25.5 million total network weights [29]. Figure 5 displays the ResNet50 model's design.

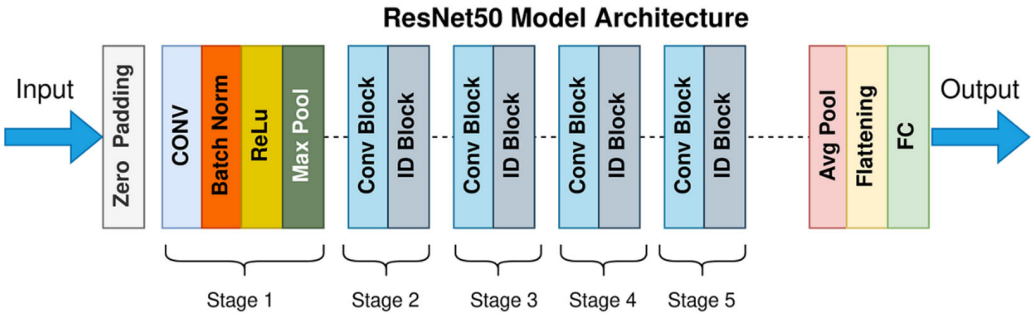


Figure 5. The fundamental design of ResNet50.

3.3. GoogLeNet Model as Feature Extractor

The GoogLeNet model was introduced by Szegedy [30] in 2016 and was designed based on the architecture of inception. Via the use of inception modules, the network is able to select among a variety of convolutional filter dimensions for every block. Since its original proposal, the inception module has received constant upgrades and improvements. There are 48 convolution layers in this architecture. The image size that it accepts as input is 224 by 224. In this study, we utilize the pretrained GoogLeNet network, extracting features from the last "FC1000" layer. Figure 6 displays the GoogLeNet model's design.

3.4. AlexNet Model as Feature Extractor

The AlexNet model was suggested by Krizhevsky et al. [31]. There are a total of 25 layers in the AlexNet model: 3 fully connected layers, 5 convolutional layers, pooling layers, and a SoftMax layer. Additionally, the ReLU activation function is incorporated. AlexNet transformed the field of machine learning and computer vision by surpassing conventional methods with its cutting-edge recognition accuracy. This discovery turned the field's history around by generating a huge spike in interest in deep learning for problems involving visual recognition and classification [32]. In this study, we extract features from AlexNet, obtaining a feature representation that is 4096-dimensional by using

both of the fully connected layers (FCLs) that are the most recent [33]. Figure 7 depicts the AlexNet design.

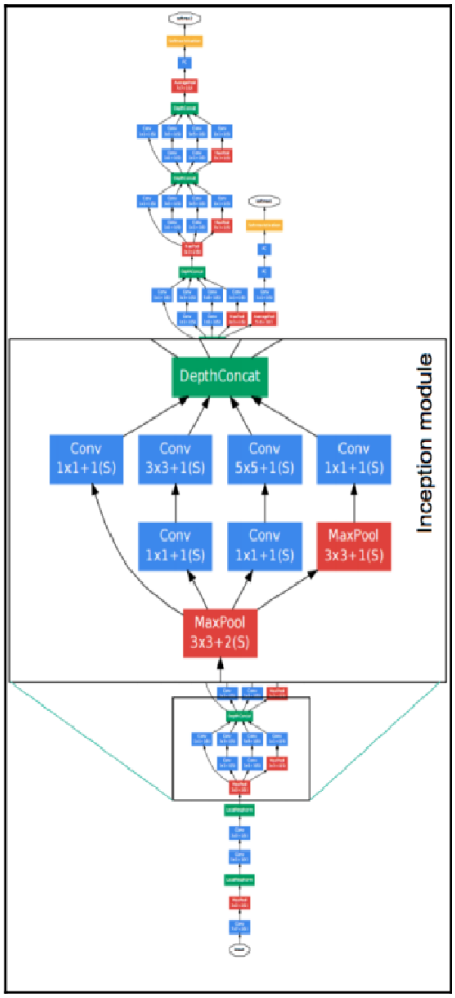


Figure 6. The fundamental design of GoogLeNet.

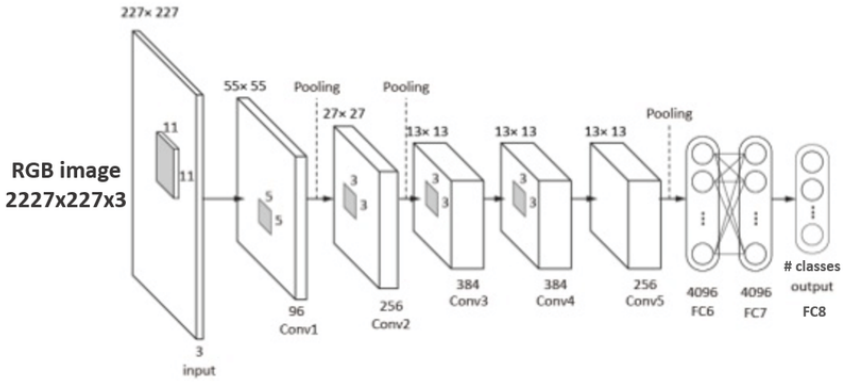


Figure 7. The fundamental design of AlexNet [32].

3.5. SqueezeNet Model as Feature Extractor

In order to create a lighter, more effective computational neural network with smaller parameters, experts at Deep Scale, University of California, Berkeley, and Stanford University designed the SqueezeNet model [34]. The fifteen layers that make up SqueezeNet are divided into five distinct levels: two layers for convolution, three layers for pooling,

eight layers for fire, one global average layer for pooling, and one output layer for Soft-Max. Because it has a smaller number of parameters, the network becomes more efficient. (2) Applications created for it are portable and need lower communication; and (3) its model size is below 5 MB, making it simple to integrate into embedded devices while maintaining a high level of accuracy. Figure 8 depicts the SqueezeNet design.

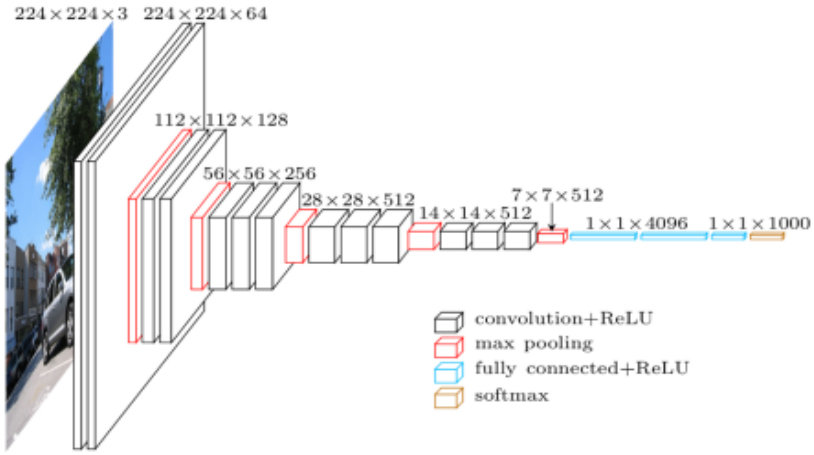


Figure 8. The fundamental design of SqueezeNet.

4. The Proposed Methods

The process used for comparison and the proposed method focuses on the four primary stages of malware detection (malware transformation, feature extraction, classification, and evaluation). Initially, we convert the malware RGB images into grayscale versions. The feature extraction process extracts the features for each grayscale image from various models such as ResNet50, GoogLeNet, AlexNet, DenseNet201, and SqueezeNet. The classification is carried out via GDA, KNN, logistic, SVM, RF, and ensemble learning. The application of assessment measures allows for the examination of a better classifier. A confusion matrix can be used to establish these measurement ratios. Figure 9 depicts a process diagram for the process for comparison and the presented method. In addition to that, Algorithm 1 was created as well. The subsections that follow contain all of the specifics for each stage.

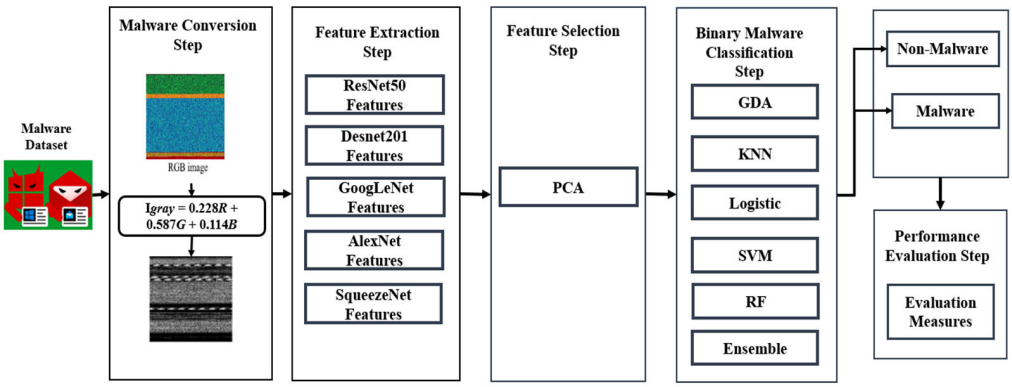


Figure 9. A general diagram regarding the proposed method and comparison process.

Algorithm 1. Process implementation pretrained models for malware detection

Input: RGB Malware Image from dataset.
Output: Non-Malware/Malware Image.
Begin
For
Step 1: Load the dataset
Step 2: Read all the images using the "Imread ()" function to read each image;
Step 3: The RGB image should be converted to the grayscale image through the use of a Matlab code called "rgb2gray ()";
Step 4: Resizing Process:
Step 4.1: For Resnet50, DenseNet201, and GoogLeNet model, Resize the converted images into (224, 224) pixels.
Step 4.2: For AlexNet and SqueezeNet model, Resize the converted images into (227, 227) pixels.
Step 5: Load pre-trained CNN models (Resnet50, DenseNet201, GoogLeNet, AlexNet, and SqueezeNet).
Step 6: Activations Process:
Step 6.1: Outputs from (" fc1000") the last convolutional layer of ResNet50.
Step 6.2: Outputs from (fc1000) the last layer of DenseNet201.
Step 6.3: Outputs from ('pool5-drop_7×7_s1') the last layer of GoogLeNet (Inception V3).
Step 6.4: Outputs from (fc7) the last layer of AlexNet.
Step 6.5: Outputs from (pool10) the last layer of SqueezeNet.
Step 7: Extract Feature Vector:
Step 7.1: For ResNet50, Store the 1000 feature vector dimension.
Step 7.2: For DenseNet201, Store the 1000-feature vector dimension.
Step 7.3: For GoogLeNet, Store the 1024-feature vector dimension.
Step 7.4: For AlexNet, Store the 4096-feature vector dimension.
Step 7.5: For SqueezeNet, Store the 1000 feature vector dimension.
Step 8: Apply feature selection method:
Step 8.1: Apply PCA method.
Step 9: Replace the fully connection layers FC with new classifier.
Step 9.1: Training Process:
A. Train the GDA classifier on the selected extracted feature vectors;
B. Train the KNN classifier on the selected extracted feature vectors;
C. Train the Logistics classifier on the selected extracted feature vectors;
D. Train the SVM classifier on the selected extracted feature vectors;
E. Train the RF classifier on the selected extracted feature vectors;
F. Train the Ensemble classifier on the selected extracted feature vectors;
Step 9.2: Testing Process:
A. The GDA model that has been trained is put through a series of tests to determine if an image contains malware or not.
B. The KNN model that has been trained is put through a series of tests to determine if an image contains malware or not.
C. The Logistics model that has been trained is put through a series of tests to determine if an image contains malware or not.
D. The SVM model that has been trained is put through a series of tests to determine if an image contains malware or not.
E. The RF model that has been trained is put through a series of tests to determine if an image contains malware or not.
F. The Ensemble model that has been trained is put through a series of tests to determine if an image contains malware or not.
End for
End

Step 1: Malware Transformation

Conti et al. [35] first introduced the binary visualization technique to portray binary information elements as grayscale images. This means that white is represented by 255, black by 0, and other numbers are intermediary grayscale shades.

In order to convert malware samples into binary images, a malware binary is taken as an 8-bit unsigned integer vector and afterwards arranged into a two-dimensional array. Despite significant variety, the visual textures of the same families share a lot of similarities [36]. Nonetheless, the MaleVis dataset's RGB byte images of the PE binary files are visible. Therefore, these RGB images need to be converted to grayscale. Figure 10 illustrates how images of malware that are RGB have been transformed into images of malware that are grayscale. Subsequently, the resizing method will modify the malware images' dimensions to fit the given input size for every pretrained CNN model. The new malware images' dimensions are 224 × 224 or 227 × 227.

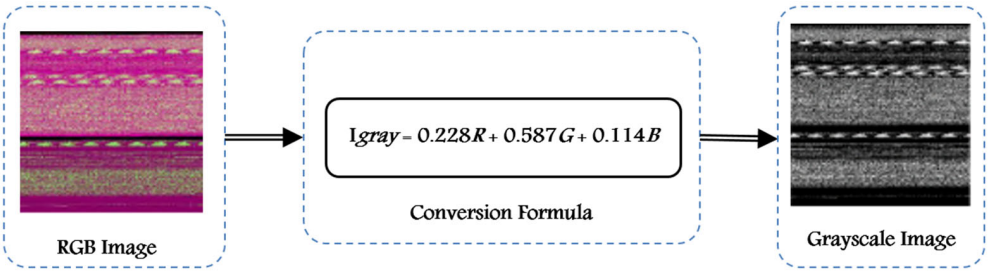


Figure 10. The graphical representation of the transformation operation [4].

Step 2: Feature Extraction

Malware classification techniques rely primarily on the feature extraction procedure. Because Pretrained models have already learned to obtain appropriate features from large amounts of data, they are employed as feature extractors for malware image detection. In this step, several models ranging in complexity and layer number are used, including ResNet50, GoogLeNet, AlexNet, DenseNet201, and SqueezeNet. These models will be used on our datasets in order to reduce the overfitting issue and determine which model is most effective at classifying images as malware. This step procedure is the input image that will be processed across the pretrained model, and the image's feature vector is obtained by using the output of the final layer before the layers are eliminated.

By using transfer learning, we can train deep learning models on small datasets without overfitting.

Step 2.1: ResNet50 Features Extraction

ResNet-50 is only utilized for feature extraction in this case. Instead of being used for classification, the network is used as a feature extractor. This indicates that the extracted feature vector is the result of the final pooling layer before the classification layer. The basic structure of ResNet50 for feature extraction is displayed in the Algorithm 2. The ResNet50 feature vector that is produced is 1 × 2048.

Algorithm 2. ResNet50 model as feature extraction.

Input: RGB Malware Image from dataset.
Output: 1 × 2048 features vector dimension.
Begin
For
1: Upload the preparation malware image.
2: Normalize each image input that falls between the 0 and 255.
3: Change the resolution to (224 × 244).
4: Load the ResNet50 model.
5: Load ResNet50 model weights.
6: The model's last fully connected (FC) layers have been eliminated or are not included.
7: Build an entirely new model for feature extraction.
8: Using ResNet50's final convolutional layer ("avg_pool" or "pool5"), the activation is produced.
9: An array has been generated from the image.
10: Reducing the features array's dimensions to one.
11: Print the flattened array.
12: Keeping the generated 2048 feature vector.
End for
End

Step 2.2: DenseNet201 Features Extraction

The network architecture of DenseNet-201 consists of several dense blocks, transition blocks, and an ultimate classification layer that will be eliminated. The purpose of the transition blocks is to minimize the feature maps' spatial dimensions in the areas between dense blocks. The basic structure of DenseNet201 for feature extraction is displayed in the Algorithm 3. The DenseNet201 feature vector that is produced is 1 × 1000 from the last fully connected layers (fc1000).

Algorithm 3. DenseNet201 model as feature extraction

Input: RGB Malware Image from dataset.
Output: 1 × 1000 features vector dimension.
Begin
For
1: Upload the preparation malware image.
2: Normalize each image input that falls between the 0 and 255.
3: Change the resolution to (224 × 244).
4: Load the DenseNet201 model.
5: Load DenseNet201 model weights.
6: The model's last fully connected (FC) layers have been eliminated or are not included.
7: Build an entirely new model for feature extraction.
8: Using **DenseNet201** final convolutional layer ("fc1000"), the activation is produced.
9: An array has been generated from the image.
10: Reducing the features array's dimensions to one.
11: Print the flattened array.
12: Keeping the generated 1000 feature vector.
End for
End

Step 2.3: GoogLeNet Features Extraction

Because it employs inception modules that minimize the number of parameters that must be learned, GoogLeNet is renowned for its parameter economy. Here, features were extracted from the final "FC1000" layer of the GoogLeNet model. The basic structure of GoogLeNet for feature extraction is displayed in Algorithm 4. The GoogLeNet feature vector that is produced is 1 × 5643.

Algorithm 4. GoogLeNet model as feature extraction.

Input: RGB Malware Image from dataset.
Output: 1 × 5643 features vector dimension.
Begin
For
1: Upload the preparation malware image.
2: Normalize each image input that falls between the 0 and 255.
3: Change the resolution to (224 × 244).
4: Load the GoogLeNet model.
5: Load GoogLeNet model weights.
6: The model's last fully connected (FC) layers have been eliminated or are not included.
7: Build an entirely new model for feature extraction.
8: Using GoogLeNet final convolutional layer ("pool5-drop_7×7_s1"), the activation is produced.
9: An array has been generated from the image.
10: Reducing the features array's dimensions to one.
11: Print the flattened array.
12: Keeping the generated 5643 feature vector.
End for
End

Step 2.4: AlexNet Features Extraction

There are fifteen layers in the AlexNet design. In order to minimize spatial dimensions, the network starts with convolutional layers and then moves on to max-pooling layers. The classification layer is then eliminated, and the final product of the convolutional layers is flattened. The basic structure of AlexNet for feature extraction is displayed in Algorithm 5. The AlexNet feature vector that is produced is 1 × 4096 from the last two fully connected layers (FCLs).

Algorithm 5. AlexNet model as feature extraction.

Input: RGB Malware Image from dataset.
Output: 1 × 4096 features vector dimension.
Begin
For
1: Upload the preparation malware image.
2: Normalize each image input that falls between the 0 and 255.
3: Change the resolution to (224 × 244).
4: Load the AlexNet model.
5: Load AlexNet model weights.
6: The model's last fully connected (FC) layers have been eliminated or are not included.
7: Build an entirely new model for feature extraction.
8: Using AlexNet final convolutional layer ("fc2"), the activation is produced.
9: An array has been generated from the image.
10: Reducing the features array's dimensions to one.
11: Print the flattened array.
12: Keeping the generated 4096 feature vector.
End for
End

Step 2.5: SqueezeNet Features Extraction

The fifteen layers that make up SqueezeNet are divided into five distinct levels: two layers for convolution, three layers for pooling, eight layers for fire, one global average layer for pooling, and an ultimate classification layer that will be eliminated. Because it has a smaller number of parameters, the network becomes more efficient. The basic structure

of SqueezeNet for feature extraction is displayed in Algorithm 6. The SqueezeNet feature vector that is produced is 1 × 1000 from the last fully connected layers (pool10).

Algorithm 6. SqueezeNet model as feature extraction.

Input: RGB Malware Image from dataset.
Output: 1 × 1000 features vector dimension.
Begin
For
1: Upload the preparation malware image.
2: Normalize each image input that falls between the 0 and 255.
3: Change the resolution to (227 × 247).
4: Load the SqueezeNet model.
5: Load SqueezeNet model weights.
6: The model's last fully connected (FC) layers have been eliminated or are not included.
7: Build an entirely new model for feature extraction.
8: Using SqueezeNet final convolutional layer ("avg_pool10"), the activation is produced.
9: An array has been generated from the image.
10: Reducing the features array's dimensions to one.
11: Print the flattened array.
12: Keeping the generated 1000 feature vector.
End for
End

Step 3: Feature Selection

By keeping useful features and eliminating duplicate and unnecessary ones, feature selection is essential to lowering the overall dimension of the dataset. Therefore, principal component analysis (PCA) is used because it is a well-known and useful approach for feature selection. PCA feature selection is a technique for reducing a dataset's total amount of parameters while keeping the most essential information. Selecting only the most significant components is the primary step in PCA, which involves transforming the original dataset into a new collection of variables that are linearly combined with the original variables [37]. PCA works in four different steps. The process involves normalizing the data, creating a covariance matrix, computing the eigenvectors and eigenvalues, and organizing the resulting eigenvectors in a descending order [38]. The pretrained models' features were retrieved from the input data, the best features were selected, and then these selected extracted features were applied in the classification step.

Step 4: Classification

For the classification step, the collected features are subsequently stored in a vector representation. To classify the malware families, a different classifier is trained using these features. A review of the literature found that several trials using various classifiers showed improved accuracy results. In the present paper, we employed GDA, KNN, logistic, SVM, RF, and ensemble learning as efficient methods for malware detection.

Step 4.1: Gaussian Discriminant Analysis (GDA) Classifier

For many classification applications, Gaussian Discriminant Analysis (GDA) [39] is a viable option because it is an extensively researched and widely understood approach. Since data can be roughly categorized using a normal distribution, the GDA approach is frequently employed for data classification. The classifier is trained using training datasets, which yield a discriminant function that indicates which class a given piece of data is more likely to belong to. If there are fewer training examples versus predicted parameters, GDA can be affected by outliers and overfit the data.

Step 4.2: K-Nearest Neighbor (KNN) Classifier

A common and easy-to-use supervised classifier is the KNN [40] model. It involves categorizing a sample according to the k samples from the training set that is closest to it. KNN with a 9-neighbor configuration was employed in this study due to the fact that if we increase K, the overfitting decreases. Although various distance functions may be selected, the Euclidean distance function is typically employed. The sample is then categorized among the k closest samples using a majority vote system or a comparable method. The benefit of KNN is its simplicity, lack of need for an explicit training phase, and high effectiveness in a wide range of applications, particularly with large training sets. The process for building a KNN model is as follows. Previous feature vectors are divided into training and testing data sets. Then, these training sets will be used to train these models. Lastly, the trained KNN model is tested to identify the class of malware that it falls into [41].

Step 4.3: The logistic regression Classifier

The most common supervised machine learning technique, logistic regression [42], represents the majority of applications for binary classification when a logistic function, sometimes referred to as a sigmoid function, is used to generate a probability value ranging from 0 to 1, based on inputs that are independent variables. Three categories exist for logistic regression: binomial, multinomial, and ordinal. Ordinary regression serves as the foundation for this logistic regression [42].

Step 4.4: SVM Classifier

Support Vector Machine (SVM) [43] is an effective machine learning algorithm used for classification and regression tasks. In most cases, however, it is utilized for addressing categorization issues in machine learning. SVMs are useful in many applications. Finding the optimal line or decision boundary to divide n-dimensional space into classes is the aim of the SVM algorithm. We refer to this optimal decision boundary as a hyperplane. To build an SVM model for malware detection, the feature vectors are partitioned into two groups called training and testing. On the training set, these models will be trained. The SVM model that has been trained to identify the class of malware that it corresponds to is tested.

Step 4.5: RF Classifier

Leo Breiman introduced the Random Forest in 2001 [44], and it plays a significant role in classification tasks. Its foundation is the idea of ensemble learning, which is the process of combining multiple classifiers in order to address a difficult problem and improve the functioning of the model. RF is a machine learning method based on decision trees. It can handle huge datasets that are highly dimensional. It improves the model's accuracy and avoids an overfitting problem [45].

Step 4.6: Extreme Learning Machines (ELM) Classifier.

The simple connections in an ELM [46] make it a design that is considered uncomplicated. It eliminates the issues of overfitting and slow training speed as compared to the traditional neural network learning technique. The empirical risk reduction theory forms the basis of ELM, and its learning process requires just one iteration. Consequently, it does a great job of forecasting performance in specific fields. Given the abundance of newly discovered malware variations, ELMs are appealing for malware classification.

5. Results and Discussion

This section is divided into four parts: public datasets, the metrics of performance evaluation, evaluation conclusions, and comparison with different methodologies. The experiment was conducted depending on some of the specifications. See Table 3 for additional details.

Table 3. Experimentation specifications description.

Hardware	Properties
PC	HP laptop (Hewlett-Packard Company, Palo Alto, CA, USA)
Operating system	Microsoft Windows 10 64-bit (OS) (Microsoft, Redmond, DC, USA)
RAM	8 GB
Processor	Intel(R) Core(TM) i7-6500U CPU @ 2.50 GHz 2.60 GHz (Intel, Santa Clara, CA, USA)
Software	MATLAB version R2020a
Graphics Card	Intel® HD Graphics 520 (NVIDIA GTX 950M) (NVIDIA, Santa Clara, CA, USA)

5.1. Datasets

The Malimg dataset [47] was employed to evaluate the effectiveness of the proposed approach. The Malimg dataset contains an imbalance of 9339 malware images, which were collected across 25 different families and classes. Table 4 shows that every sample in the dataset belongs to one of the 25 malware families. Malware images may have varying dimensions. The size and texture of these datasets are similar. A classifier can easily accomplish the task. Variants of several malware families have been shown to be extremely distinctive when compared with various other datasets.

Table 4. Summary of the Malimg dataset families.

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

An efficient malware detection software with the ability to differentiate between malware and malicious files was found. It is therefore evident that the benign files class is not included. Therefore, it is inappropriate to investigate the proposed approach using the database as it is now configured. Thus, the researcher added an additional kind of benign input to the database by gathering files that are not harmful (350 benign files) sourced from the MaleVis dataset. The Malimg dataset provides an extensive amount of train and test datasets that we can use, which makes it ideal for use in the CNN model.

5.2. Performance Evaluation Metric

Two metrics, namely detection accuracy and error, are employed to assess the proposed technique. The following formula [13] can be used to determine these measures:

1. Accuracy: the ratio of successfully classified photos to the overall number of input images, in accordance with Equation (1).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2. Error: the percentage of all images to all images that were wrongly classified, in accordance with Equation (2).

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

Figure 11 lists each term that may be ascertained using the previously provided equations.

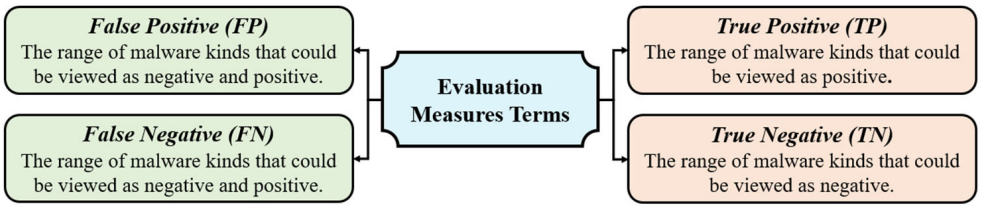


Figure 11. A detailed description of every term.

5.3. Evaluation Results

In order to guarantee the precision and reliability of the experimental results, a 10-fold cross-validation technique is employed [48]. By merging multiple different datasets, this approach minimizes bias in performance evaluation [49]. One part is used for the test set, while the other nine are selected for the training set. There are ten tests conducted in all, and the procedure is finally assessed by adding up the results of each experiment. To find the most effective classifier among a variety of pretrained model features, accuracy, and error measurements are computed. The accuracy and error results of five pretrained models feature extraction techniques and six classification techniques across the Malimg databases are displayed. To further verify the efficacy of findings, tests are carried out with the Malimg unbalanced datasets.

5.3.1. Performance Findings Using an Unbalanced Malimg Dataset

We evaluate the performance measures of the full extracted features (FEFs) and selected extracted features (SEFs) on the Malimg dataset. The FEFs are obtained from five pretrained models, including ResNet50, DenseNet201, GoogLeNet, AlexNet, and SqueezeNet. Using the PCA method, different feature dimensions, such as 50 and 500, are used to obtain the SEFs. Finally, the GDA, KNN, LOG, SVM, RF, and ELM classifiers are used to perform the classification method for binary classification (malware versus non-malware (benign)). Accuracy and error are among the metrics taken into consideration for the evaluation. These metrics offer an extensive assessment of the models' performance. The performance metrics for the FEFs and SEFs of ResNet50, DenseNet201, GoogLeNet, AlexNet, and SqueezeNet features are summarized in Tables 5–9, respectively.

A selection of assessment criteria, such as accuracy and error, is made in order to compare all classifiers. Evidently, the bold stands for the highest values.

Table 5. Evaluation measure results of ResNet50 features and a number of different classifiers over the Malimg dataset.

Classifier	Feature Extracted Dimension					
	FEF_1000F		SEF_50F		SEF_500F	
	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)
GDA	95.13	4.87	96.15	3.65	94.24	5.76
KNN	95.90	4.10	96.54	3.46	95.77	4.93
LOG	95.52	4.48	96.16	3.84	91.04	8.96
SVM	94.49	5.51	96.29	3.71	95.39	4.61
RF	95.65	4.35	96.03	3.97	95.01	4.99
ELM	71.59	28.41	95.89	4.11	95.77	4.23

Table 6. Evaluation measure results of DenseNet201features and a number of different classifiers over the Malimg dataset.

Classifier	Feature Extracted Dimension					
	FEF_1000F		SEF_50F		SEF_500F	
	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)
GDA	95.65	4.35	96.54	3.46	95.90	4.10
KNN	95.77	4.23	96.93	3.07	95.26	4.74
LOG	92.69	7.31	96.29	3.71	92.06	7.94
SVM	95.13	4.87	96.16	3.84	95.00	5.00
RF	96.29	3.71	96.67	3.72	95.20	4.80
ELM	70.42	29.58	93.85	6.15	94.37	5.63

Table 7. Evaluation measure results of GoogLeNet features and a number of different classifiers over the Malimg dataset.

Classifier	Feature Extracted Dimension					
	FEF_1024F		SEF_50F		SEF_500F	
	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)
GDA	96.41	3.59	97.06	2.94	96.03	3.97
KNN	94.49	5.51	95.90	4.10	95.02	4.98
LOG	95.01	4.99	96.15	3.85	95.65	4.35
SVM	95.52	4.48	96.03	3.97	95.13	4.87
RF	94.11	5.89	95.77	4.23	94.49	5.51
ELM	66.45	33.55	95.52	4.48	95.38	4.62

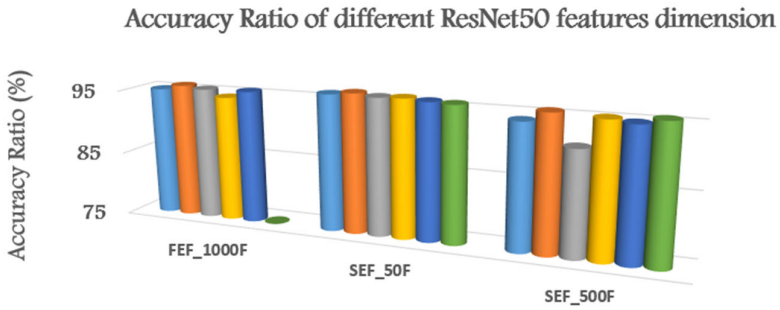
Table 8. Evaluation measure results of AlexNet features and a number of different classifiers over the Malimg dataset.

Classifier	Feature Extracted Dimension					
	FEF_4096F		SEF_50F		SEF_500F	
	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)
GDA	87.32	12.68	96.54	3.46	94.62	5.38
KNN	95.72	4.28	96.64	3.36	95.39	4.61
LOG	95.64	4.36	96.16	3.84	94.11	5.89
SVM	93.60	6.40	91.90	8.10	89.63	10.37
RF	94.75	5.25	95.65	4.35	94.25	5.75
ELM	61.84	38.46	90.78	9.22	89.88	10.12

Table 9. Evaluation measure results of SqueezeNet features and a number of different classifiers over the Malimg dataset.

Classifier	Feature Extracted Dimension					
	FEF_1000F		SEF_50F		SEF_500F	
	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)	Accuracy Rate (%)	Error Rate (%)
GDA	94.24	5.76	96.20	3.80	93.73	6.27
KNN	94.88	5.12	96.67	3.33	95.13	4.87
LOG	94.75	5.25	96.54	3.46	89.76	10.24
SVM	95.65	4.35	96.05	3.95	94.88	5.12
RF	95.03	4.97	96.16	3.84	95.77	4.23
ELM	93.60	6.40	95.01	4.99	94.18	5.82

The findings can be examined in terms of the optimal feature and classifier, and this will be discussed in the next section. We decided to start at the feature dimension of 50 and stop at the feature dimension of 500 based on the findings. We chose the FEFs and SEFs of the ResNet50 model as examples. As can be seen in Figure 12, there is no requirement for us to proceed above size 500 because the accuracy ratio at (SEF_50F) increased (96.54%) but decreased until it reached (95.77%) at size 500. Furthermore, we can observe that the accuracy ratio at FEF_1000F is 95.90%, but the accuracy is 96.54% when we utilize SEF_50F. This demonstrates how feature selection helps enhance performance. When comparing the accuracy of five pretrained models to various feature selection dimensions, SEF_50F consistently obtains superior accuracy. Therefore, we considered SEF_50F for carrying out additional experiments for the purpose of this paper.



	FEF_1000F	SEF_50F	SEF_500F
GDA	95.13	96.15	94.24
KNN	95.9	96.54	95.77
LOG	95.52	96.16	91.04
SVM	94.49	96.29	95.39
RF	95.65	96.03	95.01
ELM	71.59	95.89	95.77

Figure 12. Accuracy percentage for various ResNet50 feature dimensions under the Malimg dataset.

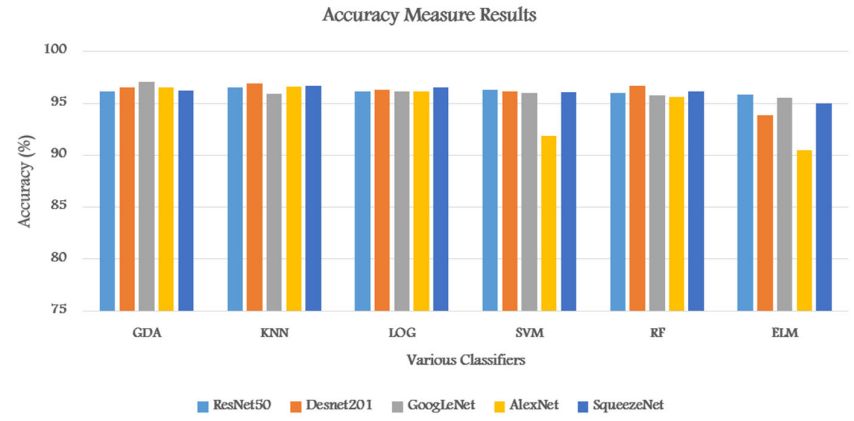
Performance Analysis of Each Classifier

An analysis of the findings in Tables 5–9 will make selecting the optimal classifier possible. With the purpose of applying the assessment metrics, a comparison across all classifiers will be carried out, including accuracy and error. One of the things that we are conducting right now is determining the overall mean for each metric for all features for each classifier that we are currently testing.

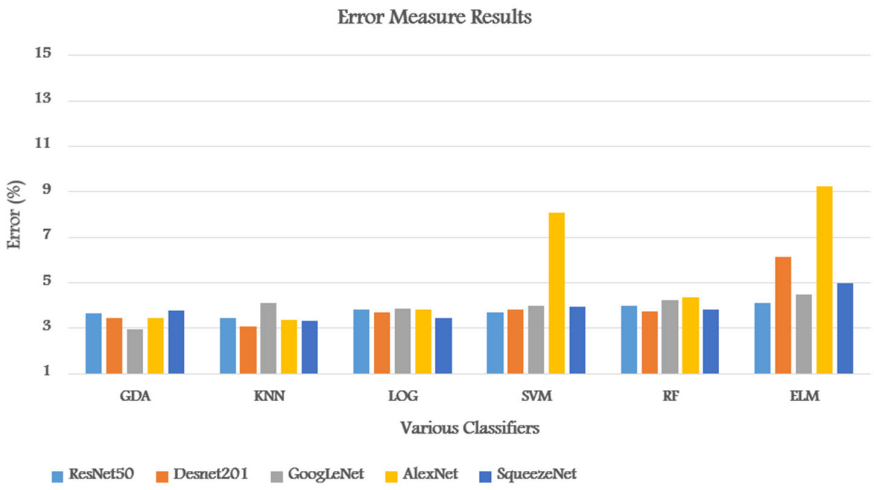
It was discovered that the average accuracy values for KNN, GDA, LOG, SVM, RF, and ELM are 96.49, 96.53, 96.26, 95.28, 96.05, and 94.15, respectively. When compared to all of the classifiers, the KNN classifier was able to obtain the maximum accuracy rate in regard

to the ResNet50, DenseNet201, GoogLeNet, AlexNet, and SqueezeNet features, as can be seen in Figure 13a. It was discovered that the average error values for KNN, GDA, LOG, SVM, RF, and ELM are 3.46, 3.46, 3.74, 4.71, 4.02, and 5.79, respectively. When compared to all of the classifiers, the KNN classifier was able to obtain the minimum error rate in regard to the ResNet50, DenseNet201, GoogLeNet, AlexNet, and SqueezeNet features, as can be seen in Figure 13b.

When compared to the other classifiers, the KNN classifier demonstrated superior performance in terms of both accuracy and error. Then, the GDA classifier, on the other hand, is ranked in the second position, while the ELM classifier is ranked at last place and generates the lowest possible results. The KNN classifier produced the highest results over two assessment criteria, as indicated by the previous results obtained from a number of other classifiers that were applied to the Malimg dataset. The KNN classifier was therefore taken into consideration for the purpose of carrying out more tests in this paper.



(**a**)



(**b**)

Figure 13. Comparative analysis of several classifiers evaluation metrics using the Malimg dataset: (**a**) accuracy; (**b**) error.

Performance Analysis of Each Feature

An analysis of the findings in Tables 5–9 will make selecting the optimal feature possible. With the purpose of applying the assessment metrics, a comparison across all features will be carried out, including accuracy and error. One of the things that we are carrying out right now is determining the overall mean for every metric for all of the classifier for each feature that we are currently testing.

It was discovered that the average accuracy values for ResNet50, DenseNet201, GoogLeNet, AlexNet, and SqueezeNet features are 96.17, 96.073, 96.071, 94.56, and 96.10, respectively. When compared to all of the models' features, the DenseNet201 feature model was able to obtain the maximum accuracy rate in regard to the KNN, GDA, LOG, SVM, RF, and ELM classifiers, as can be seen in Figure 14a. It was discovered that the average error values for ResNet50, DenseNet201, GoogLeNet, AlexNet, and SqueezeNet features are 3.79, 3.99, 3.92, 5.38, and 3.89, respectively. When compared to all of the models' features, the DenseNet201 feature model was able to obtain the minimum error rate in regard to the KNN, GDA, LOG, SVM, RF, and ELM classifiers, as can be seen in Figure 14b.

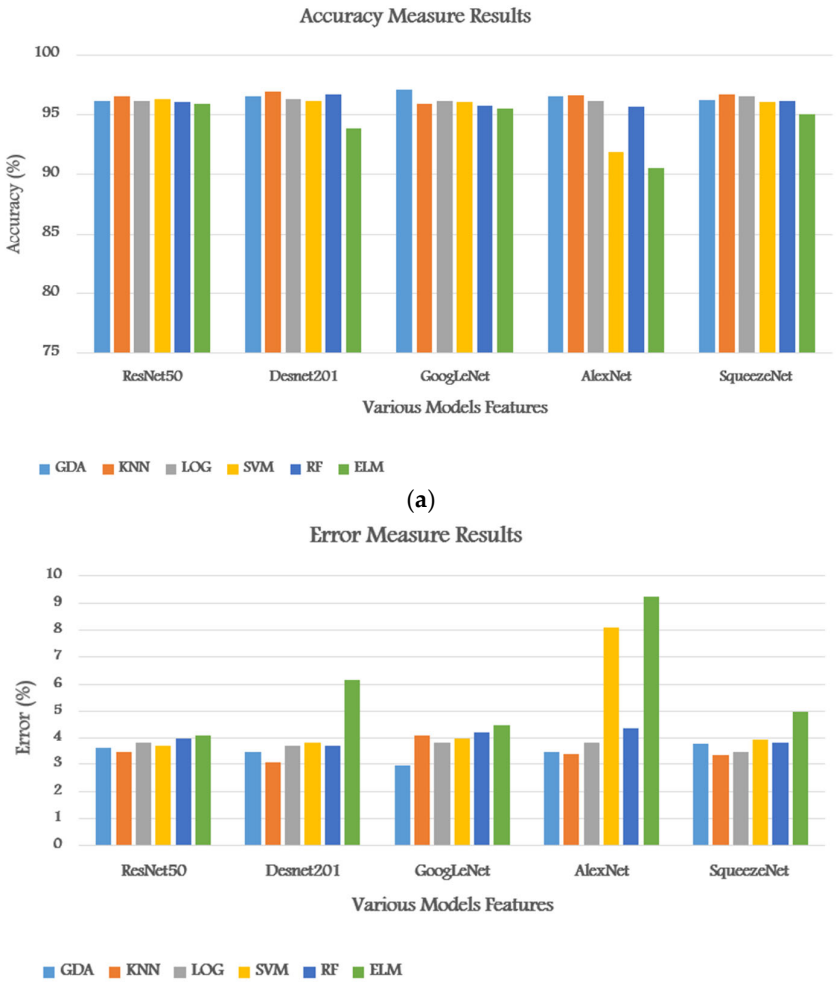


Figure 14. Comparative analysis of several models features evaluation metrics using Malimg dataset: (**a**) accuracy; (**b**) error.

When compared to the other models' features, the DenseNet201 feature demonstrated superior performance in terms of both accuracy and error. Then, the GoogLeNet feature model, on the other hand, ranks in the second position, while the SqueezeNet feature ranks last and generates the lowest possible results. DenseNet201 produced the highest results over two assessment criteria, as indicated by the previous results obtained from a number of other model features that were applied to the Malimg dataset. The DenseNet201 feature was therefore taken into consideration for the purpose of carrying out more tests in this paper.

The proposed DenseNet201-KNN produced the highest results over two assessment criteria, as indicated by the previous results obtained from a number of other classifiers and models' features that were applied to the Malimg dataset. The proposed DenseNet201-KNN was therefore taken into consideration for the purpose of carrying out more tests in

this paper. In addition, the feature vector dimension of the proposed method is 1×50. By reducing the dimensionality of features, the approach minimizes computation complexity, maintains speed, and optimizes memory usage.

5.4. Comparative Results of Current Methods

In order to evaluate both the recommended DenseNet201-KNN method effectiveness, we compare them to other methods in the following subsection. For most of these techniques, one of the most important steps is the extraction of features from pretrained CNN models. Table 10 displays the results of the comparison. Table 10 compares a number of different malware classification methods with the DenseNet201-KNN method. Therefore, we make use of the results of methods that make use of the Malimg databases, varied feature domains, and multiple previously trained models, as well as feature extraction and classification methods.

Table 10. Comparing performance with previous approaches.

Author/Year	Feature Extraction Method	Classifier	Dataset	Accuracy (%)
Makandar and Patrot, 2017 [50]	Gabor wavelet	KNN	Malimg	89
Rezende et al., 2017 [51]	ResNet-50	SoftMax	Malimg	98
Hsien-De et al., 2018 [52]	LBP	SVM	Malimg	75
Khan et al., 2018 [14]	GoogLeNet	SoftMax		74
Hashemi et al.,2019 [53]	LBP	KNN	Malimg	91
N. Bhodia et al., 2019 [54]	ResNet-50	KNN	Malimg	94
Naeem et al., 2020 [55]	VGG16	DCNN	Malimg	97
Vasan et al., 2020 [17]	VGG16, RESNET50, INCPETIONV3	CNN	Malimg	97
Mohammed et al., 2021 [56]	DCT	CNN	MaleVis	96
Sharif et al., 2023 [57]	ResNet-50	CNN	Malimg	81
DenseNet201-KNN (Proposed)	DenseNet201	KNN	Malimg	96

A performance assessment of the presented method in comparison with other popular methods utilizing the Malimg dataset is displayed in Figure 15, utilizing percent accuracy. It is evident from the graphic that the recommended technique produces results that are almost the same but with a smaller feature vector dimension. Table 10 illustrates how the suggested feature extraction method employs fewer feature vector sizes, making it computationally quicker than other existing methods.

The methods described in [17,51] are extremely accurate. However, as Table 10 shows, an integration of deep model features (VGG16, RESNET50, and INCPETIONV3) generates a huge feature vector dimension, so more time was needed.

The accuracy results from the studies [51,55] can be considered similar to the highest degree of accuracy of the method that is being presented. These methods, however, depend on the features of the ResNet-50 and VGG16 models, which need large feature vector dimensions. Because of its magnitude, the feature vector requires a huge computational expenditure. However, our technique only needed a 1×50 feature vector.

Methods for detecting malware [50] utilize the transform domain. The approach in [56] is based on the discrete cosine transform (DCT). Nevertheless, as Table 10 illustrates, they have the disadvantage of taking an excessively long time. Alternative malware detection approaches [14,50,52,57] perform badly in comparison to our proposed method.

The proposed detection accuracy improved to 96% by incorporating DenseNet201 and KNN, beating alternative methods.

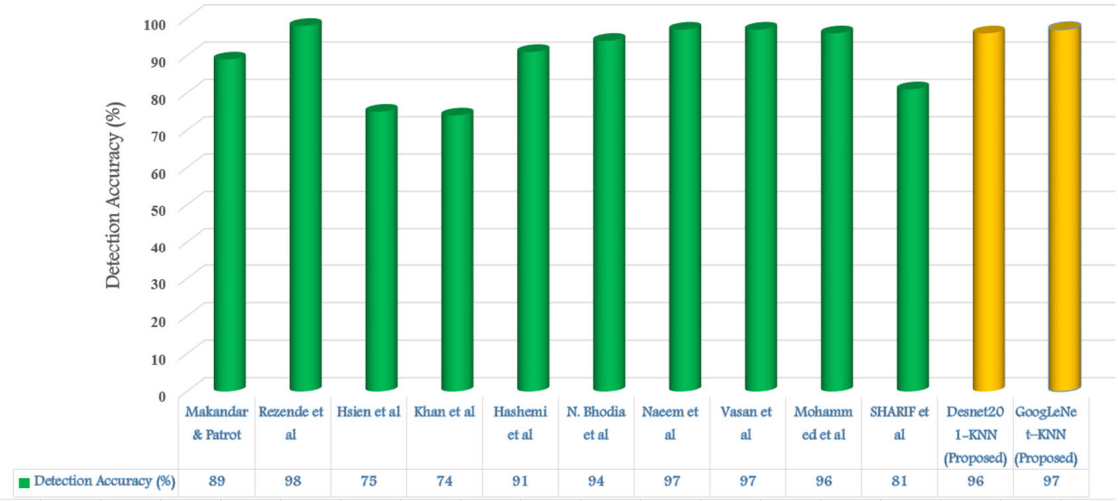


Figure 15. Performance evaluation comparison of different malware detection methods.

6. Conclusions

The main aim of the proposed is to identify and determine the best features and classifiers and then use it in order to propose the malware detection method. According to the comparison findings, KNN is the best machine learning classifier. It outperformed the other classifiers on the Malimg datasets in terms of both accuracy and error. The GDA classifier came in second place. The ELM classifier comes in last and yields the worst results. In addition, DenseNet201 is the best pretrained model under the Malimg dataset. Therefore, the proposed DenseNet201-KNN method had an accuracy rate of 96% and a minimal error rate of 3.07%. The proposed methods surpass existing state-of-the-art approaches. The proposed feature extraction is computationally quicker than most other methods since it uses a lightweight design and fewer feature vector dimensions. Therefore, the proposed method might work well for implementation on small electronic devices. Nonetheless, additional space is required to increase accuracy. Therefore, in the future, a comparison study of a wide range of deep learning models will be needed. It is feasible to lessen the issue of imbalanced datasets and improve the efficacy of the proposed malware detection technique by balancing and expanding the dataset.

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

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in the study are openly available in [47].

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

References

1. 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.

2. Hammad, B.T.; Jamil, N.; Rusli, M.E.; Z'Aba, M.R.; Ahmed, I.T. Implementation of lightweight cryptographic primitives. *J. Theor. Appl. Inf. Technol.* **2017**, *95*, 5126–5141.
3. Bayer, U.; Moser, A.; Kruegel, C.; Kirda, E. Dynamic analysis of malicious code. *J. Comput. Virol.* **2006**, *2*, 67–77. [CrossRef]
4. Ahmed, I.T.; Jamil, N.; Din, M.M.; Hammad, B.T. Binary and Multi-Class Malware Threads Classification. *Appl. Sci.* **2022**, *12*, 12528. [CrossRef]
5. Da'u, A.; Salim, N. Recommendation system based on deep learning methods: A systematic review and new directions. *Artif. Intell. Rev.* **2020**, *53*, 2709–2748. [CrossRef]
6. 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.
7. 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]
8. Pant, D.; Bista, R. Image-based Malware Classification using Deep Convolutional Neural Network and Transfer Learning. In Proceedings of the 3rd International Conference on Advanced Information Science and System, Sanya, China, 26–28 November 2021; pp. 1–6.
9. Kumar, S.; Janet, B. DTMIC: Deep transfer learning for malware image classification. *J. Inf. Secur. Appl.* **2022**, *64*, 103063. [CrossRef]
10. Gyamfi, N.K.; Goranin, N.; Čeponis, D.; Čenys, A. Malware detection using convolutional neural network, a deep learning framework: Comparative analysis. *J. Internet Serv. Inf. Secur.* **2022**, *12*, 102–115. [CrossRef]
11. Asam, M.; Hussain, S.J.; Mohatram, M.; Khan, S.H.; Jamal, T.; Zafar, A.; Khan, A.; Ali, M.U.; Zahoora, U. Detection of exceptional malware variants using deep boosted feature spaces and machine learning. *Appl. Sci.* **2021**, *11*, 10464. [CrossRef]
12. Aslan, Ö.; Yilmaz, A.A. A new malware classification framework based on deep learning algorithms. *IEEE Access* **2021**, *9*, 87936–87951. [CrossRef]
13. Hammad, B.T.; Jamil, N.; Ahmed, I.T.; Zain, Z.M.; Basheer, S. Robust Malware Family Classification Using Effective Features and Classifiers. *Appl. Sci.* **2022**, *12*, 7877. [CrossRef]
14. 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]
15. Lo, W.W.; Yang, X.; Wang, Y. An xception convolutional neural network for malware classification with transfer learning. In Proceedings of the 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), IEEE, Canary Islands, Spain, 24–26 June 2019; pp. 1–5.
16. Singh, A.; Handa, A.; Kumar, N.; Shukla, S.K. Malware classification using image representation. In Proceedings of the Cyber Security Cryptography and Machine Learning: Third International Symposium, CSCML 2019, Beer-Sheva, Israel, 27–28 June 2019; pp. 75–92.
17. 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]
18. Marastoni, N.; Giacobazzi, R.; Dalla Preda, M. Data augmentation and transfer learning to classify malware images in a deep learning context. *J. Comput. Virol. Hacking Tech.* **2021**, *17*, 279–297. [CrossRef]
19. Anandhi, V.; Vinod, P.; Menon, V.G. Malware visualization and detection using DenseNets. *Pers. Ubiquitous Comput.* **2021**, *28*, 153–169. [CrossRef]
20. Asam, M.; Khan, S.H.; Akbar, A.; Bibi, S.; Jamal, T.; Khan, A.; Ghafoor, U.; Bhutta, M.R. IoT malware detection architecture using a novel channel boosted and squeezed CNN. *Sci. Rep.* **2022**, *12*, 15498. [CrossRef] [PubMed]
21. Shaukat, K.; Luo, S.; Varadharajan, V. A novel deep learning-based approach for malware detection. *Eng. Appl. Artif. Intell.* **2023**, *122*, 106030. [CrossRef]
22. Dawra, B.; Chauhan, A.N.; Rani, R.; Dev, A.; Bansal, P.; Sharma, A. Malware Classification using Deep Learning Techniques. In Proceedings of the 2023 2nd Edition of IEEE Delhi Section Flagship Conference (DELCON), Rajpura, India, 24–26 February 2023; pp. 1–7.
23. 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.
24. 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.
25. Wang, S.-H.; Zhang, Y.-D. DenseNet-201-based deep neural network with composite learning factor and precomputation for multiple sclerosis classification. *ACM Trans. Multimed. Comput. Commun. Appl.* **2020**, *16*, 1–19. [CrossRef]
26. Qu, L.; Wu, C.; Zou, L. 3D dense separated convolution module for volumetric medical image analysis. *Appl. Sci.* **2020**, *10*, 485. [CrossRef]
27. Theckedath, D.; Sedamkar, R.R. Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks. *SN Comput. Sci.* **2020**, *1*, 79. [CrossRef]
28. 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.

29. Alzubaidi, L.; Zhang, J.; Humaidi, A.J.; Al-Dujaili, A.; Duan, Y.; Al-Shamma, O.; Santamaría, J.; Fadhel, M.A.; Al-Amidie, M.; Farhan, L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *J. Big Data* **2021**, *8*, 53. [CrossRef]
30. Yoo, H.-J. Deep convolution neural networks in computer vision: A review. *IEIE Trans. Smart Process. Comput.* **2015**, *4*, 35–43. [CrossRef]
31. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
32. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; Van Esesn, B.C.; Awwal, A.A.S.; Asari, V.K. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv* **2018**, arXiv:1803.01164.
33. Ahmed, I.T.; Hammad, B.T.; Jamil, N. Effective Deep Features for Image Splicing Detection. In Proceedings of the 2021 IEEE 11th International Conference on System Engineering and Technology (ICSET), Shah Alam, Malaysia, 6 November 2021; pp. 189–193.
34. Wang, S.; Kang, B.; Ma, J.; Zeng, X.; Xiao, M.; Guo, J.; Cai, M.; Yang, J.; Li, Y.; Meng, X. A deep learning algorithm using CT images to screen for Corona Virus Disease (COVID-19). *Eur. Radiol.* **2021**, *31*, 6096–6104. [CrossRef] [PubMed]
35. Conti, G.; Dean, E.; Sinda, M.; Sangster, B. Visual reverse engineering of binary and data files. In Proceedings of the International Workshop on Visualization for Computer Security, Cambridge, MA, USA, 15 September 2008; pp. 1–17.
36. Nataraj, L.; Yegneswaran, V.; Porras, P.; Zhang, J. A comparative assessment of malware classification using binary texture analysis and dynamic analysis. In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, Chicago, IL USA, 21 October 2011; pp. 21–30.
37. Omuya, E.O.; Okeyo, G.O.; Kimwele, M.W. Feature selection for classification using principal component analysis and information gain. *Expert Syst. Appl.* **2021**, *174*, 114765. [CrossRef]
38. Wang, S.; Yu, X.; Jia, W. A new population initialization of particle swarm optimization method based on pca for feature selection. *J. Big Data* **2021**, *3*, 1. [CrossRef]
39. Sharifi, K.; Leon-Garcia, A. Estimation of shape parameter for generalized Gaussian distributions in subband decompositions of video. *IEEE Trans. Circuits Syst. Video Technol.* **1995**, *5*, 52–56. [CrossRef]
40. 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]
41. Ahmed, I.T.; Hammad, B.T.; Jamil, N. Forgery detection algorithm based on texture features. *Indones. J. Electr. Eng. Comput. Sci.* **2021**, *24*, 226–235. [CrossRef]
42. Wilson, J.R.; Lorenz, K.A. *Modeling Binary Correlated Responses Using SAS, SPSS and R*; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9.
43. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4.
44. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
45. Ahmed, I.T.; Hammad, B.T.; Jamil, N. Common Gabor Features for Image Watermarking Identification. *Appl. Sci.* **2021**, *11*, 8308. [CrossRef]
46. 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 (IEEE Cat. No. 04CH37541), Budapest, Hungary, 25–29 July 2004; Volume 2, pp. 985–990.
47. 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.
48. Ahmed, I.T.; Hammad, B.T.; Jamil, N. Image Steganalysis based on Pretrained Convolutional Neural Networks. In Proceedings of the 2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA), Selangor, Malaysia, 12 May 2022; pp. 283–286.
49. Ahmed, I.T.; Der, C.S.; Jamil, N.; Hammad, B.T. Analysis of Probability Density Functions in Existing No-Reference Image Quality Assessment Algorithm for Contrast-Distorted Images. In Proceedings of the 2019 IEEE 10th Control and System Graduate Research Colloquium (ICSGRC), Shah Alam, Malaysia, 2–3 August 2019; pp. 133–137.
50. 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.
51. 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), IEEE, Cancun, Mexico, 18–21 December 2017; pp. 1011–1014.
52. 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), Shanghai, China, 15–17 January 2018; pp. 2633–2642.
53. Hashemi, H.; Hamzeh, A. Visual malware detection using local malicious pattern. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 1–14. [CrossRef]
54. Bhodia, N.; Prajapati, P.; Di Troia, F.; Stamp, M. Transfer learning for image-based malware classification. *arXiv* **2019**, arXiv:1903.11551.
55. 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]

56. Mohammed, T.M.; Nataraj, L.; Chikkagoudar, S.; Chandrasekaran, S.; Manjunath, B.S. Malware detection using frequency domain-based image visualization and deep learning. *arXiv* **2021**, arXiv:2101.10578.
57. Sharif, H.U.; Jiwani, N.; Gupta, K.; Mohammed, M.A.; Ansari, M.F. A deep learning based technique for the classification of malware images. *J. Theor. Appl. Inf. Technol.* **2023**, *101*, 135–160.