applied sciences

MDPI

Article

A New Framework for Visual Classification of Multi-Channel Malware Based on Transfer Learning

Zilin Zhao, Shumian Yang * and Dawei Zhao *

Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China
* Correspondence: yangshm@sdas.org (S.Y.); zhaodw@sdas.org (D.Z.)

Abstract: With the continuous development and popularization of the Internet, there has been an increasing number of network security problems appearing. Among them, the rapid growth in the number of malware and the emergence of variants have seriously affected the security of the Internet. Traditional malware detection methods require heavy feature engineering, which seriously affects the efficiency of detection. Existing deep-learning-based malware detection methods have problems such as poor generalization ability and long training time. Therefore, we propose a malware classification method based on transfer learning for multi-channel image vision features and ResNet convolutional neural networks. Firstly, the features of malware samples are extracted and converted into grayscale images of three different types. Then, the grayscale image sizes are processed using the bilinear interpolation algorithm to make them uniform in size. Finally, the three grayscale images are synthesized into three-dimensional RGB images, and the RGB images processed using data enhancement are used for training and classification. For the classification model, we used the previous ImageNet dataset (>10 million) and trained all the parameters of ResNet after loading the weights. For the evaluations, an experiment was conducted using the Microsoft BIG benchmark dataset. The experimental results showed that the accuracy on the Microsoft dataset reached 99.99%. We found that our proposed method can better extract the texture features of malware, effectively improve the accuracy and detection efficiency, and outperform the compared models on all performance metrics.

Keywords: network security; deep learning; transfer learning; convolutional neural networks; malware classification

check for updates

Citation: Zhao, Z.; Yang, S.; Zhao, D. A New Framework for Visual Classification of Multi-Channel Malware Based on Transfer Learning. *Appl. Sci.* **2023**, *13*, 2484. https://doi.org/10.3390/app13042484

Academic Editor: Juan Pavón

Received: 28 December 2022
Revised: 8 February 2023
Accepted: 13 February 2023
Published: 15 February 2023

1. Introduction

Currently, with the rapid development of Internet technology, people enjoy the convenience brought by the Internet, but this also brings network security risks. In 1981, the first computer virus known to be widely spread (ELK Cloner) was born. ELK Cloner was not destructive at the time, but was designed as a good-natured joke. However, with over time, the number of malware has been increasing every year, and many variants have been created. At the same time, the harmfulness of malware is also increasing. Malware can be installed and run on computers or terminal devices without the consent of users, thus harming the legitimate rights and interests of users.

According to the China Internet Network Security Monitoring Data Analysis Report, in the first half of 2021, the number of malicious program samples captured was about 23.07 million, and the daily average number of transmissions reached more than 5.82 million, involving about 208,000 malicious program families, and more than 866,000 new mobile Internet malicious programs were found through independent capture and vendor exchange. The number of hosts infected with computer malicious programs in China was about 4.46 million, up 46.8% year-on-year. There are many ways for malware to invade or spread. Attackers can take advantage of vulnerabilities in web services, browsers, and operating systems or use social engineering technology to allow end users run malicious code to spread malicious software. In addition, attackers also use some obfuscation technologies, such as code integration, dead-code insertion, subroutine reordering, instruction substitution, and code transposition, to circumvent the detection by traditional defense means (such as firewalls, anti-virus software, and gateways) [1]. Moreover, this affects manual analysis to some extent. Therefore, how to detect and classify malware quickly and accurately is a research hotspot in the field of network security.

Malware detection technologies are generally divided into two categories: static [2–5] and dynamic [6–9] analysis methods. The static analysis methods do not require actual running of malware samples and focus on code analysis, for example analyzing the structure and syntax of malicious binary files and performing classification detection according to family characteristics. Dynamic analysis methods can reveal the behavior of malware, which is not affected by obfuscation technology and can detect unknown malware samples. However, this is very time-consuming, and the operation steps are very cumbersome. Although the above problems can be solved by using static analysis methods, they rely on efficient anti-virus engines and comprehensive virus databases, resulting in the lower accuracy of sample detection.

In order to compensate for the shortcomings of static analysis methods, relevant personnel began to investigate the malware detection and classification methods based on visualization [10,11] and have achieved good performance. Despite the increasing number and variety of malware, the core of the binary or assembly code of malware in the same family has similarities. After visualization, the essential characteristics of image texture and structure will not be changed, which can effectively combat the problem of malware confusion. Machine-learning-based visualization techniques [12,13] are simple to operate, short in consumption time, and less dependent on data. However, they are vulnerable to the impact of feature engineering and have difficulty dealing with large volumes of malware. Although significant numbers of malware samples can be identified daily, it is still very difficult to detect and classify malware solely by relying on manual work.

In recent years, deep learning [14–17] has developed rapidly in detecting malware samples, avoiding the complexity of the manually extraction of features in traditional machine learning. At the same time, it reduces the impact on the accuracy of malware detection caused by analysts' lack of experience and ability and can effectively resist malware attacks. Therefore, some researchers began to combine deep learning with visualization technologies [18–23] to classify and detect malware, providing a new way to detect and classify malware. Currently, most malware visualization methods convert the information in binary files into grayscale images. Then, the similarity between the texture features or other malware features is observed to determine if it is the same family. Finally, deep learning methods are combined to improve malware detection accuracy. However, visualization techniques based on currently available deep neural networks still face many challenges. For example, the convolutional neural network model used is too simple, with insufficient generalization ability, and cannot effectively extract important information contained in malware. At the same time, the extracted malware features are single and only focus on one aspect of the malware features, which affects the accuracy of classification and detection. The generated image generally needs to be uniform in size, which easily causes information loss in the process of processing and contains obvious noise features.

To solve the above problems, this paper proposes a malicious software detection and classification method based on depth learning of multi-channel RGB images. This method transforms malware into a three-channel image, combines the depth convolution neural network model with transfer learning, integrates bilinear interpolation, multi-channel feature extraction, data enhancement, and other technologies, and improves the efficiency and accuracy of classification detection based on effectively retaining malware sample information. The main contributions of this paper are summarized as follows:

- A new method of malware visualization is proposed, which transforms the extracted malware binary file information into three different grayscale images and fuses them into three-channel RGB images. We can analyze malware from a multi-dimensional perspective and effectively retain relevant information in the malware binary file.
- We propose a new framework for malware classification using convolutional neural networks. We combined the ResNet34 convolutional neural network model with the model trained on the ImageNet dataset for transfer learning and compared the performance with other ResNet network models. This method does not require reverse analysis and can achieve a good training effect in a short time, effectively improve the accuracy of malware classification, and enhance the model's generalization ability.
- Compared with the sample images' interception or filling method, this paper used image interpolation algorithms to deal with the size of grayscale images and compared and analyzed the performance of different interpolation algorithms. Without loss of image information, it can effectively avoid the loss of feature information.
- The RGB images generated were processed by using the contrast limited adaptive histogram equalization (CLAHE) data enhancement method, which can better deal with the problem of data imbalance. At the same time, it can effectively limit noise amplification, expand the local contrast, and display more details of the smooth areas.

The rest of the study is organized as follows: Section 2 introduces the traditional malware detection methods and related research on using visualization techniques to classify malware. Section 3 introduces in detail the visual characteristics of multi-channel RGB images based on transfer learning and the new framework of the malware classification method based on the ResNet convolutional neural network. Section 4 gives the experimental results and makes a comparative analysis of the experimental data. Finally, Section 5 summarizes the study and provides some ideas and suggestions.

2. Related Work

The wide spreading of malware constitutes a serious threat. Effective classification and detection of malware are a key focus of current research. Generally, static or dynamic analysis methods are used to extract malware features, and relevant algorithms are used to classify the extracted features into families. In recent years, more and more researchers have begun to use visual methods to process malware samples and combine machine learning or deep learning to classify and detect malware samples. Therefore, this section will mainly review the following three related technologies: methods based on static features, methods based on dynamic features, and feature methods based on visualization technology.

2.1. Methods Based on Static Features

The static analysis method does not require the actual execution of malware samples and has the advantages of a simple operation and low time complexity. Ding et al. [24] proposed a method based on control flow to represent malware binary files as operation code streams. First, the control flow graph is generated, and all possible execution paths are obtained by traversing the graph. Then, the n-gram method is used to extract data features, and the feature based on opcodes is selected by information gain and document frequency. Finally, three classifiers, KNN, C4.5, and SVM, were trained to classify and detect malware samples. Shalaginov et al. [25] first extracted and analyzed various static features of 32-bit Windows malware binary files. Then, different machine learning technologies were used for the classification. The final experimental results showed that C4.5 and KNN had better performance effects, while naive Bayes had the worst classification effect.

Gibert et al. [26] proposed a baseline system based on the deep learning method, which is composed of manual design and is end-to-end, including API components, mnemonic components, byte components, and feature fusion and classification components. The malware detection and classification tasks are addressed by extracting features using static analysis methods on the Microsoft malware dataset and combine different types of features to find the relationship between different modalities. Wu et al. [27] proposed a new malware classification method based on the graph and graph-embedded network function, which combines structural information and semantic information to generate feature vectors of binary files for malware analysis. Combining static analysis methods and deep learning, Kakisim et al. [28] proposed a sequential-opcode-embedding-based malware classification method. Each malware is represented as an opcode subgraph structure, and the random walk method and word embedding technology are used to automatically learn the opcodes.

2.2. Methods Based on Dynamic Features

Dynamic analysis methods generally execute malware samples in a virtual environment, which can trace the actual running path of the malware and is not affected by obfuscation technology. Bonfante et al. [29] constructed an efficient signature matching engine based on tree automation technology, combined syntax, semantic analysis, and a combined control flow graph to classify malware samples. Compared with the strategy proposed by Christodorescu et al. [30] and Bruschi et al. [31], this experiment paid more attention to the efficiency of the optimization algorithm. The experimental dataset selected 10,156 malware samples and 2653 benign programs. The experimental results showed that a false positive rate of 0.1% was achieved by using a fully automated method to extract the signatures. Lin et al. [32] proposed a malware classification method based on a virtual time controller. By accelerating sandboxes and using virtual time control mechanisms and information measurement methods, we can more efficiently observe the dynamic behavior of target software in the virtual environment and detect malware.

Sun et al. [33] proposed a malware classification method based on a classification behavior graph. They executed malware samples in the cuckoo sandbox, studied the behavior pattern of the malware using the analysis report of the JSON files, and extracted API calls from the JSON files. Next, they constructed a behavior graph based on the extracted API information to classify the malware. Amer et al. [34] built a multi-perspective malware detection model by calling API sequences to inherit statistics, context, and image mining functions. Li et al. [35] designed a novel graphical neural network model, which also uses API call sequences. Composed of API feature engineering and API call graph learning, the dynamic malware analysis framework can effectively solve the malware detection and malware type classification problems.

2.3. Methods Based on Visualization Techniques' Features

The visualization of malware can observe its texture characteristics more intuitively and can reduce the impact of obfuscation technology. Therefore, more and more related research based on visualization technology has emerged. In recent years, the combination of machine learning and visualization technology has gradually begun to rise. Nataraj et al. [36] proposed for the first time applying visual methods to detect malware samples and use machine learning methods for classification. First, the malware binary sequence is read. Then, it is cut into 8-bit subsequences and converted to 0–255 pixel intervals through a decimal system. Finally, according to the file size, the image width is fixed, and the gray image is generated. The core of the malicious binary file is represented in the image features. Liu et al. [37] proposed a visual detection method for malware based on machine learning. First, the malware is transformed into a feature image. Then, the size is unified by scaling technology, and the AE and AT technologies are combined. Finally, the machine learning method is used to detect the image samples.

In order to overcome the deficiencies of machine learning, such as being vulnerable to feature engineering, it is suitable to train small sample datasets. At present, visualization technology based on deep learning has become popular. Ni et al. [38] used the SimHash technology to convert malware binary files into grayscale images and then identified their families through convolutional neural networks. The experimental results showed that the accuracy rate on the Microsoft malware dataset was as high as 99.26%, and a new sample can be recognized on average in 1.41 s. Zhao et al. [39] proposed a malware classification method based on visualization and feature fusion. By extracting bytes and operation codes from the Microsoft dataset, they combined global structure information with local semantic information and used convolutional neural networks to classify and detect the generated images. Conti et al. [40] extracted three different features from malware binary files to generate images (grayscale images, Markov images, and entropy images) and fused them into three-channel GEM images. Experiments were conducted using two different small sample learning architectures, the convolutional Siamese neural network (CSNN) and shallow few-shot (Shallow-FS). The experimental results showed that the CSNN is suitable for scarce data; otherwise, Shallow-FS is better.

In order to speed up the model training and improve the performance of classification detection, researchers have introduced the method of transfer learning. Vasan et al. [41] transformed malware binary files into color images and used data enhancement methods to process unbalanced datasets in the fine-tuning process. The parameters of the pre-trained model on the ImageNet dataset were applied to the ResNet network model to classify and detect the generated images. Chaganti et al. [42] proposed an EfficientNetB1 network based on transfer learning to classify byte-level images of malware. Three different methods for image representation of malware were studied, and the optimal image representation of malware with a fixed image width was studied in more detail. Although good results were achieved, the different types of grayscale images generated were not fused.

3. Methodology

To improve the accuracy of malware classification, we propose a malware classification method using multi-channel image visual characteristics and a convolutional neural network, which is based on transfer learning. It includes three components: feature extraction, the generation of multi-channel images, and the construction of convolutional neural networks for classification and detection. The framework of our method is shown in Figure 1. Firstly, the malware binary files are converted into three different grayscale images, and the grayscale images are zoomed in and out using a bilinear interpolation algorithm to unify the size. Then, they are fused into three-dimensional RGB images as feature images, and the CLAHE algorithm is used to enhance the synthesized feature image. Finally, the transfer learning method is combined with the improved convolutional neural network to classify and detect malware samples.

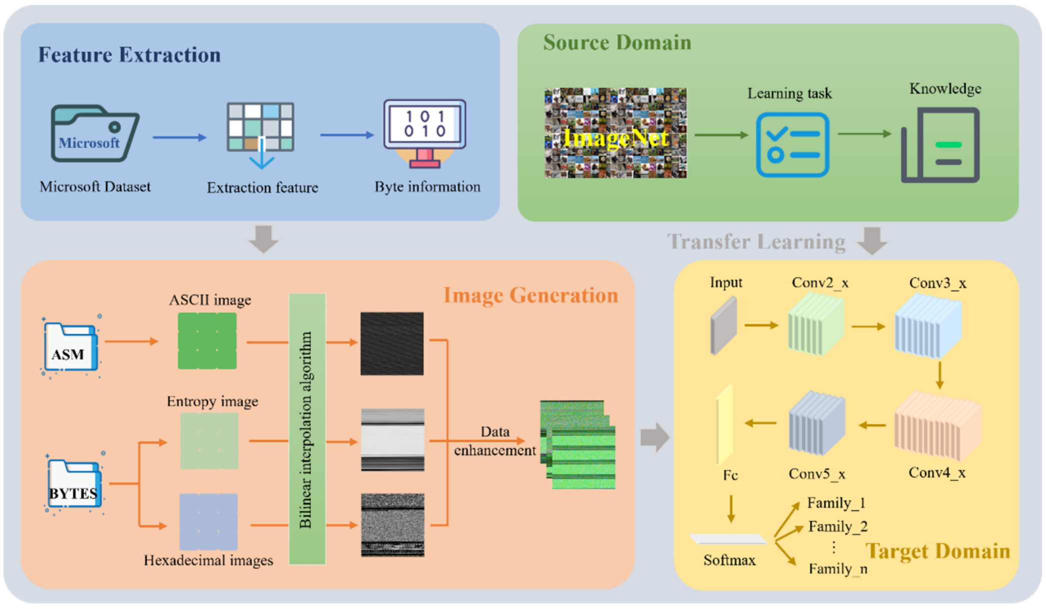

Figure 1. The proposed program framework.

3.1. Image Representation of Malware

Generally, the color depth of the points in a black-and-white image is called grayscale, and the range is 0–255 (0 represents black; 255 represents white). Therefore, black and white images are also grayscale images. To generate feature images, it is necessary to convert the information in the malware file. In this paper, we propose three conversion methods to generate grayscale images. ASCII images are generated using asm files, and hexadecimal and entropy images use byte files. This is demonstrated using four malware samples, generating different grayscale images, as shown in Figure 2, where each rectangle (image) in Ramint (a), Remmit (b), and Remmit (c) has a one-to-one correspondence; they are generated by the same malware using three different methods.

3.1.1. ASCII Images

The original byte in the malware binary file can be converted into an 8-bit gray pixel. The byte sequence is segmented by a fixed width and converted into a two-dimensional gray matrix. Firstly, we converted the byte sequence into consecutive ASCII visible characters. Then, the width of the fixed generated image was 256 pixels, so the first 256 bytes in the binary file were taken as the first row of pixels, and the bytes between 257 and 512 were taken as the second row of pixels, and so on. When the byte sequence of the last line was less than 256 bytes, 0×00 was used for filling. Finally, the height of the generated ASCII image depends on the size of the malware sample file. Figure 2a shows the generated malware ASCII image.

3.1.2. Hexadecimal Images

We converted the hexadecimal value in the malware binary file into a grayscale pixel value (in the range 0–255) for every 8 bits, and the output image was saved in PNG format. In this paper, different sizes of intervals were divided according to the sample size of the dataset used, and the results are shown in Table 1. For example, when the size of the malware sample file was less than 10 KB, the image's width was set to 64 pixels, and the height was set according to the sample file size. When the last line was less than the length of the set width, it was filled with 0×00. The generated hexadecimal image of the malware is shown in Figure 2b.

Table 1. Image width for various file sizes.

File Size	Image Pixel	File Size	Image Pixel
<10 KB	32	100 KB∼200 KB	384
10 KB∼30 KB	64	200 KB∼500 KB	612
30 KB∼60 KB	128	500 KB∼1000 KB	864
60 KB∼100 KB	256	>1000 KB	1024

3.1.3. Entropy Images

By calculating the entropy of the malware sample file to generate grayscale images, the entropy can represent the confusion degree of the byte values in the sample file. It can reflect the similarity of the sample file. By observing the generated pixel values, we found that the entropy values had a small change range. This may be the reason why we magnified entropy values. We refer to the way Fu et al. [43] generated the entropy images. The entropy is calculated as shown in Equation (1):

$$Entropy = (-\sum_{l=0}^{255} P(c_i)\log_2 P(c_i)) \times \frac{255}{8} \tag{1}$$

where $P(c_i)$ represents the probability of the occurrence of byte value i. When the byte values in the intercepted byte sequence are the same, the entropy value is 0; otherwise, the entropy value is 8. To facilitate the observation of the generated entropy images, we scaled the entropy values from 0 to 255. Figure 2c shows the generated malware entropy image.

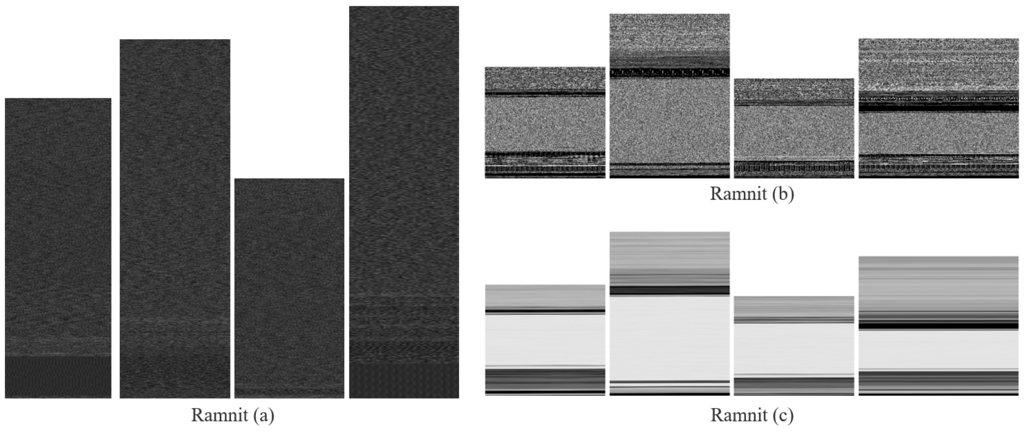

Figure 2. Generated grayscale images of malware using three different methods.

3.2. Interpolation Algorithms

When converting malware binary files to grayscale images, the size of the generated image specifications is not uniform. Therefore, we needed to standardize the generated grayscale images. To avoid the loss of information and retain the features of the sample file to the greatest extent, we used interpolation algorithms to scale the grayscale images. In the following, we will briefly introduce the nearest neighbor interpolation algorithm, bilinear interpolation algorithm, and bicubic interpolation algorithm. In the Section 4, we will compare and analyze the three interpolation algorithms through experiments and select the best interpolation algorithm to process the ASCII image, hexadecimal image, and entropy image so that they are unified into 256 $\times$ 256 sizes. The grayscale images processed by different interpolation algorithms are shown in Figure 3.

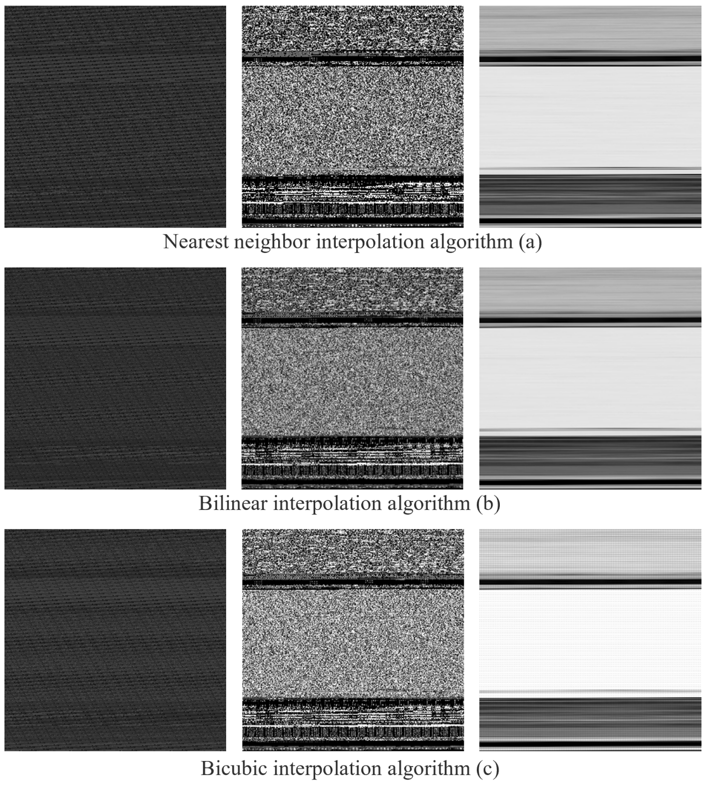
Nearest neighbor interpolation algorithm (a)
Bilinear interpolation algorithm (b)
Bicubic interpolation algorithm (c)

Figure 3. Grayscale images processed by different interpolation algorithms.

3.2.1. Nearest Neighbor Interpolation Algorithm

The nearest neighbor interpolation algorithm is simple in operation and requires less computation. Generally, the gray value of the pixel closest to the location of the sampling point to be measured is selected as the gray value of the sampling point. The generated image is shown in Figure 3a. Let point $A(x, y)$ be the sampling point to be measured and (x_0,y_0), (x_0, y_1), (x_1, y_0), and (x_1, y_1) be the pixel coordinate points of the original image. Because the (x_0, y_1) coordinate point is closest to the sampling point to be measured, so the interpolation result is $A = f(x, y) = f(x_0, y_1)$. Although the nearest neighbor interpolation algorithm is very fast, the effect is not good and will produce obvious sawtooth and mosaic phenomena.

3.2.2. Bilinear Interpolation Algorithm

The bilinear interpolation algorithm can overcome the defects of the nearest neighbor interpolation algorithm; the effect is better, but the operation speed is slightly slower. Calculate the linear interpolation of two variables, *B* and *C*, respectively, in the *x* direction. Then, conduct the linear interpolation in the y direction and obtain the sampling point to be measured through four adjacent points. Its core idea is linear interpolation in two directions, and the generated image is shown in Figure 3b. The linear interpolation formula in the *x*-direction is:

$$f(B) \approx \frac{x_1 - x}{x_1 - x_0} f(x_0, y_0) + \frac{x - x_0}{x_1 - x_0} f(x_1, y_0) \tag{2}$$

$$f(C) \approx \frac{x_1 - x}{x_1 - x_0} f(x_0, y_1) + \frac{x - x_0}{x_1 - x_0} f(x_1, y_1) \tag{3}$$

The linear interpolation formula in the y-direction is:

$$f(x, y) \approx \frac{y_1 - y}{y_1 - y_0} f(B) + \frac{y - y_0}{y_1 - y_0} f(C) \tag{4}$$

Finally, the interpolation result $f(x, y)$ obtained by using the bilinear interpolation algorithm is obtained:

$$\begin{aligned} f(x, y) \approx & \frac{(x_1 - x)(y_1 - y)}{(x_1 - x_0)(y_1 - y_0)} f(x_0, y_0) + \frac{(x - x_0)(y_1 - y)}{(x_1 - x_0)(y_1 - y_0)} f(x_1, y_0) \\ & + \frac{(x_1 - x)(y - y_0)}{(x_1 - x_0)(y_1 - y_0)} f(x_0, y_1) + \frac{(x - x_0)(y - y_0)}{(x_1 - x_0)(y_1 - y_0)} f(x_1, y_1) \end{aligned} \tag{5}$$

3.2.3. Bicubic Interpolation Algorithm

The bicubic interpolation, also known as cubic convolution interpolation, is the most-complex algorithm, with a large amount of computation and a long running time. However, it can produce smoother edges and achieve good results. The algorithm uses the gray values of sixteen points around the sampling point to perform the cubic interpolation, which considers the influence of the four adjacent points' gray values and the influence of the change rate of the gray values. The generated image is shown in Figure 3c, and the formula for calculating the pixel value $f(x, y)$ is as follows:

$$f(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} f(x_i, y_j) W(x - x_i) W(y - y_j) \tag{6}$$

$$W(x) = \begin{cases} (a+2) \mid x \mid^3 - (a+3) \mid x \mid^3 + 1, \mid x \mid \leq 1 \\ a \mid x \mid^3 - 5a \mid x \mid^2 + 8a \mid x \mid - 4a, 1 < \mid x \mid < 2 \\ 0. \qquad\qquad\qquad\qquad other \end{cases} \tag{7}$$

where $W(x)$ is used to calculate the pixel point's weight and the weight's value is related to the distance between the pixel points.

3.3. Contrast Limited Adaptive Histogram Equalization

Through the linear interpolation algorithm, the grayscale images generated by three different methods were zoomed in and out to generate 256 × 256 size images. In the next step, we performed the fusion operation. The ASCII images were used as the first channel, the entropy images as the second channel, and the hexadecimal images as the third channel to form three-dimensional RGB images. Then, the CLAHE algorithm was used to enhance the generated RGB image. At first, the image was partitioned into sub-regions of equal size, and the histogram of each block was calculated. Then, define a threshold β, and if the histogram of a block exceeds the defined threshold, the histogram is trimmed from the top, and the portion exceeding the threshold is evenly distributed to each gray level until equalization is achieved. Finally, the center point of each sub-region was used as the reference point to obtain the gray value, which was then the interpolation algorithm used to process it to obtain the enhanced image. The CLAHE algorithm is an improvement of the adaptive histogram equalization algorithm, which can enhance the image's contrast while suppressing noise. The three-channel RGB images processed using the CLAHE algorithm are shown in Figure 4, and the formula for the threshold β is shown below:

$$\beta = \frac{P}{Q}(1 + \frac{\alpha}{100}(y_{max} - 1)) \tag{8}$$

where P denotes the number of pixels in different blocks and Q is the gray levels; α represents the truncation factor, ranging from 0 to 100; y_{max} represents the maximum allowed slope, which determines the amplitude of contrast enhancement.

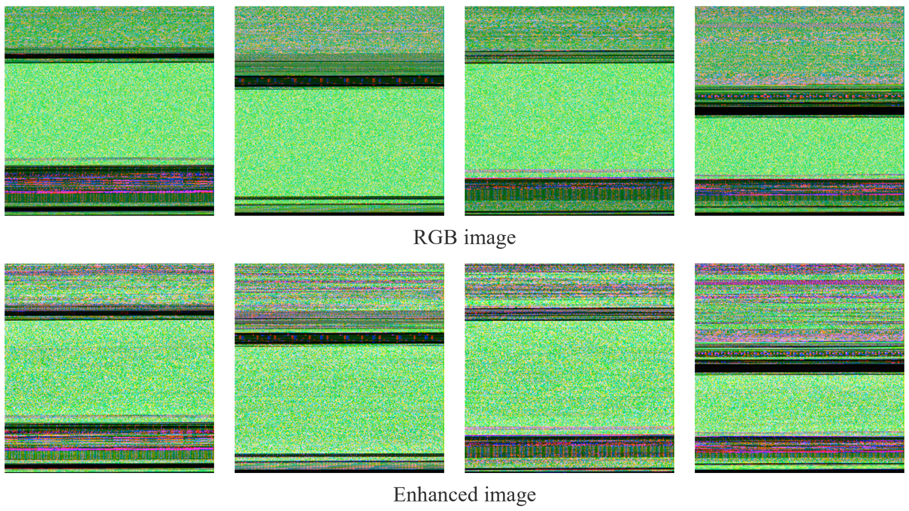

Figure 4. Images before and after processing with the data enhancement method.

3.4. Transfer Learning

Training a new convolutional neural network may require iterating dozens of epochs to achieve good training accuracy. At the same time, if the convolutional neural network is very complex, the number of parameters is very large and the dataset is very small, it will not be sufficient to train the whole network. Thus, overfitting can occur and affect the accuracy of classification detection. Therefore, we used the transfer learning method to solve the above problems. The use of transfer learning can quickly train an ideal result in a short time, and it is suitable for small datasets. Using the pre-trained model parameters on the ImageNet dataset, we applied them to our convolutional neural network model. This is equivalent to taking the model developed for Task A as the initial point and reusing the relevant parameters of Task A in the model developed for Task B.

At present, there are three common transfer learning methods: training all parameters after loading weights, only training the last several layers of parameters after loading weights, and adding a fully connected layer based on the original network model after loading weights and training only the last fully connected layer. This paper adopted the method of training all parameters after loading weights, and the classification and detection results were better than the other two methods. First, all the pre-trained model parameters were loaded into the ResNet34 convolutional neural network. Then, the network parameters of all layers in the ResNet34 convolutional neural network were trained according to the dataset used. Finally, we modified the number of nodes in the last fully connected layer.

3.5. Convolution Neural Network Classification

Theoretically, the deeper the layers of the neural network are, the more complex the model structure is, the more comprehensive the features extracted will be, and the better the effect will be. However, the effect is not good in practical applications. At the same time, the problem of gradient disappearance or gradient explosion becomes more serious when the convolutional neural network is stacked to a certain depth. Therefore, we used the ResNet convolutional neural network to classify malware images to prevent the above problems and obtain better classification and detection results. ResNet was first proposed by Microsoft Lab and was the champion of the ImageNet competition in 2015. The network proposes a residual network structure and builds an ultra-deep network structure (over 1000 layers). At the same time, it uses batch normalization to speed up the training (discard the dropout). Compared with other convolutional neural networks, ResNet can greatly improve the system's robustness and significantly improve the effect of classification detection. Figure 5 shows the proposed convolutional neural network architecture.

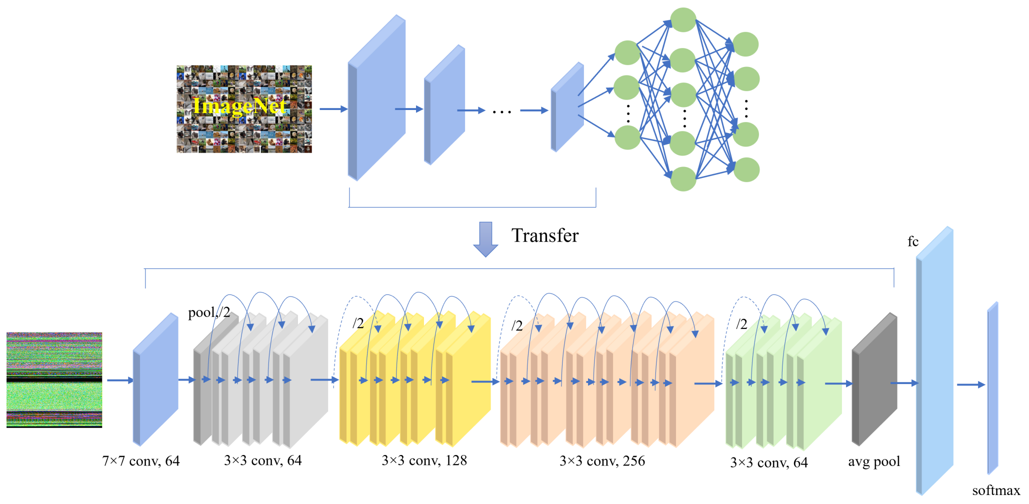

Figure 5. Proposed neural network architecture.

We took a multi-channel RGB image of size 256 × 256 as the input and the improved ResNet34 convolution neural network model was used for classification detection. First, input the image into a 7 × 7 convolutional layer with the size of the convolutional kernel, then through four convolution groups, including 16 residual network structures (3 dotted line residual network structures and 13 solid line residual network structures). Then, the downsampling of the first convolution group is realized by a maximum pooling layer with a 3 × 3 convolution core size. The remaining convolution group's downsampling is realized by the residual blocks adjacent to the previous convolution group. Finally, we output through the average pooling layer and the fully connected layer and used the softmax function to convert our output into a probability distribution. In the training phase, the convolutional neural network model was trained using the cross-entropy loss function, and the model parameters were learned using the ADAM optimizer. The formula of the cross-entropy loss function is as follows:

$$CrossEntropyLoss = -\sum_{i-1}^{N} p(x_i)\log p(x_i) + (1 - p(x_i))\log(1 - p(x_i)) \tag{9}$$

where N is the number of categories, $p(x_i)$ represents the true distribution of the samples, and $q(x_i)$ denotes the distribution predicted by the model.

Compared with the ResNet34 convolutional neural network model, we set the steps of the first and second convolutional layers of the dashed residual network structure to 1 and 2, respectively. The GELU activation function was used, and the model parameters trained on ImageNet were used for the transfer training. Therefore, this can accelerate the training speed and enhance the model's generalization ability. At the same time, the classification speed was significantly improved, and the accuracy would be better.

4. Experimental Evaluation

4.1. Dataset Statistics and Preprocessing

We evaluated the experiment using a publicly available Microsoft dataset, which was made public in 2015, with a size of nearly 0.5 TB. Each malware file has an identifier, a hash value, and a class label. Each of the raw data has a hexadecimal representation of the binary content. The title is not included to ensure sterility. The dataset has two different types of data: one is byte format, and the other is asm format. As the dataset has already been processed, it can be used directly. We converted the bytes in the binary file sample into images. The dataset contains 10,868 malware samples from nine families. The distribution of malware samples in the Microsoft dataset is shown in Table 2.

Table 2. Number of samples for each malware family in the Microsoft dataset.

Class ID	Family	Samples	Percentage
1	Ramnit	1541	0.1418
2	Lollipop	2478	0.2280
3	Kelihos_Ver3	2942	0.2707
4	Vundo	475	0.0437
5	Simda	42	0.0039
6	Tracur	751	0.0691
7	Kelihos_Ver1	398	0.0366
8	Obfuscator.ACY	1228	0.1130
9	Gatak	1013	0.0932

We used the Python programming language to implement various programs in the experiment, using the Python library to call functions and using Keras 2.10.0 and Pytorch 1.12.1 as the backend. The hardware environment included: processor: Intel i7-12500H, 12 cores, 16 threads, single display: GTX3060, video memory: 4G. We divided the dataset as follows: 90% for training and 10% for testing. The experimental parameters were set as follows: the learning rate was set to 1×10^{-4}, the batch size to 64, and the training period to 10.

4.2. Evaluation Indicators

First, we needed to understand the concepts of true positives (TPs), true negatives (TNs), false positives (FPs) and false negatives (FNs). A TP a predicted malware sample that is also a malware sample in fact. An FP is a predicted malware sample, but it is a benign software sample. An FN is a predicted benign software sample, but is actually a malware sample. A TN is a predicted benign software sample that is also a benign software sample in fact. To evaluate the performance of the classifier, the accuracy, precision, recall and F-score were used as the evaluation indicators. At the same time, the macro average and micro average were also used as indicators to detect malware categories in order to evaluate the performance comprehensively.

The accuracy is the proportion of all predictions that are correct to the total, and the formula is as follows:

$$Accuracy(A) = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

The precision is the proportion of samples correctly predicted as malware to all samples predicted as malware, and the formula is as follows:

$$Precision(P) = \frac{TP}{TP + FP} \tag{11}$$

The recall is the percentage of correctly predicted malware samples to all actual malware samples, and the formula is as follows:

$$Recall(R) = \frac{TP}{TP + FN} \tag{12}$$

The F-score is the harmonic average of precision and recall, and the formula is as follows:

$$F - score(F) = \frac{P \times R}{P + R} \times 2 \tag{13}$$

4.3. Experimental Results

4.3.1. Comparison of Malware Classification Performance with Different Dataset Division Ratios

We explored the impact of different partitioning criteria on the malware samples' classification and detection performance by dividing the dataset into different training and test set ratios. In total, we conducted a comparative analysis of four different partitioning criteria, with the training and test sets divided into ratios of 9:1, 7.5:2.5, 6:4, and 5:5. The experimental results using different division ratios to divide the dataset are shown in Table 3. By conducting the experimental analysis, we can observe that the best results were achieved when the ratio of the training set to the test set being 9:1.

Table 3. Dividing the dataset using different division ratios.

Proportion	Accuracy	Precision	Recall	F-Score
9:1	0.9999	0.9934	0.9936	0.9935
7.5:2.5	0.9959	0.9917	0.9911	0.9914
6:4	0.9921	0.9908	0.9928	0.9918
5:5	0.9908	0.9905	0.9897	0.9901

4.3.2. Comparison of Interpolation Algorithms for Classification Performance of Malware Samples

Malware binary files need to be uniform in size when they are converted into grayscale images. The experiment used the nearest neighbor interpolation algorithm (nearest curve), the bilinear interpolation algorithm (proposed curve), and the bicubic interpolation algorithm (bicubic curve) for processing. Figures 6 and 7 show the classification performances of different interpolation algorithms for the malware images. The abscissa represents the training period (our training period was set to 10), and the ordinate represents the accuracy or loss value. The experimental results showed that using the bilinear interpolation algorithm to process the images achieved better results, with a maximum accuracy of 99.99%. We observed that, although the nearest neighbor interpolation algorithm should achieve poorer results in theory, we can observe that its accuracy was only inferior to the bilinear interpolation algorithm through Figure 6. Although the bicubic interpolation algorithm performs 16-point sampling, the features obtained should be better, but it did not show a leading trend in the experiment. At the same time, with the increase of the epochs, it can be observed that the loss effect of the bilinear interpolation algorithm was obviously better than that of the other two interpolation algorithms. Therefore, this experiment used a bilinear interpolation algorithm to scale the grayscale images.

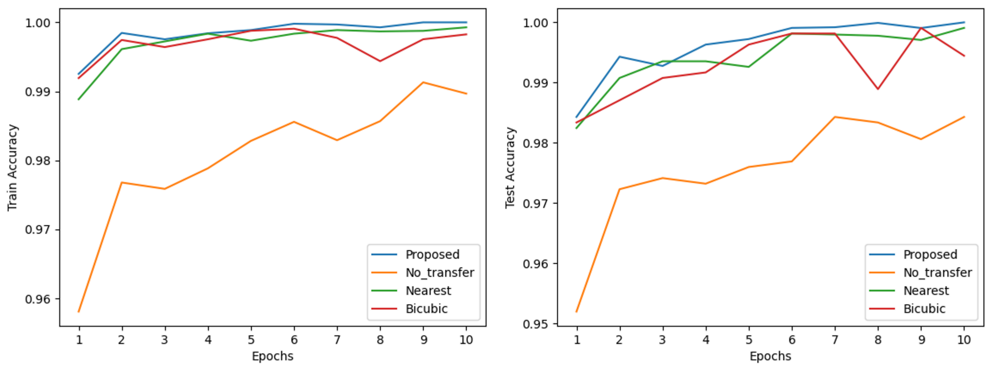

Figure 6. The changing trend of accuracy with training epochs.

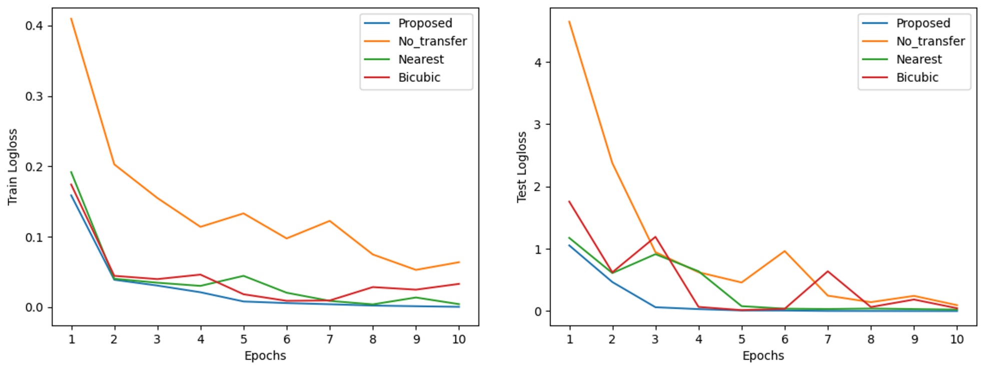

Figure 7. The changing trend of loss value with training epochs.

4.3.3. Comparison of Transfer Learning for Classification Performance of Malware Samples

In this paper, the model parameters were trained on the ImageNet dataset and all loaded into the ResNet34 convolutional neural network. After loading the weights, all parameters were trained, and we modified the number of nodes in the last layer of the fully connected layers. The experiment compared the convolutional neural networks without transfer learning with those using transfer learning. The accuracy of the training and test set changed with the epochs, as shown in Figure 6, and the loss value of the training and test set changed with the epochs, as shown in Figure 7. The experimental results showed that the ResNet34 convolutional neural network using transfer learning converged faster. With the increase of the epochs, the accuracy improved steadily, fluctuating between 0.99 and 1.0, and good performance was achieved. After just one epoch, the accuracy of the training set reached 99.25%, and the accuracy of the test set reached 98.43%, while the loss effect of the training and test sets was also greater. Compared to the ResNet34 convolutional neural network without transfer learning, the final accuracy improvement was about 0.78 percentage points, and the loss value was improved by about 6.08 percentage points. The ability to identify certain features was already in place through transfer learning, and good results could still be achieved even when applied to other areas.

4.3.4. Comparison of Image Enhancement for Malware Sample Classification Performance

Due to the unbalanced distribution of the samples in the malware family, the generated multi-channel images were processed using the CLAHE algorithm, which performs the data enhancement. The original data were compared with the enhanced data, as shown in Table 4. We found that the accuracy of the data enhancement method reached 99.99%, which was 0.10% higher than the original data. At the same time, we observed the macro average and micro average of the F-score before and after using the enhancement method, and the results achieved were significantly better. Therefore, the experimental results showed that data enhancement can reduce the dependence of the model on feature attributes, while reducing the problem of the poor detection effect due to the imbalance of the data samples.

Table 4. Comparison of results before and after using data enhancement methods.

	Accuracy	**Micro Avg**	**Macro Avg**
Proposed	0.9999	0.9935	0.9935
No-enhanced	0.9989	0.9890	0.9898

4.3.5. Comparison of Grayscale and Multi-Channel Images for Classification Performance of Malware Samples

The bytes in malware binary files were extracted, and three different methods were used to generate the grayscale images (ASCII, hexadecimal, and entropy images). Then, the bilinear interpolation algorithm was used to unify the image size. The three grayscale images were fused to generate multi-channel images. The experiment compared and analyzed the evaluation indicators (precision, recall, and F-score) of the three size-unified grayscale and multi-channel images, as shown in Figures 8–10. The abscissa represents the class label in the malware family, and the ordinate represents the value of the evaluation index. Through observation, the effect of fusion into multi-channel images was obviously due to the grayscale image of a single channel. Even when the number of samples was small, the performance was good.

In addition to comparing the evaluation indicators, we used the confusion matrix to conduct a comparative analysis of the grayscale and multi-channel images, as shown in Figure 11. Figure 11a–d correspond to the ASCII, entropy, hexadecimal, and images generated using this paper's method. We found that the image generation method proposed in this paper had a better effect, and the effect in each category was better than the other methods.

4.3.6. Comparison of Different Classification Models for Classification Performance of Malware Samples

In this paper, the classification performance of different ResNet networks for malware multi-channel images was also compared and analyzed. The experimental results are shown in Table 5. When the training cycle was set to 10, because the proposed method is based on transfer learning, it achieved a good effect quickly. However, after many experiments, the performance of convolutional neural networks without transfer learning is very poor. Therefore, we set the training period of the convolutional neural network without transfer learning to 30 to compare the classification performance of different classification models for malware samples. By observing Table 5, we find that, basically, all network models reached a high accuracy rate, which was above 96%. ResNext50_32x4d and ResNext101_32x8d are improved versions of ResNet, mainly updating the blocks. However, compared with ResNet50 and ResNet101, the accuracy rate and other indicators did not perform well. At the same time, Table 5 also compares the layers of different ResNet networks. With the increase of the layers, the performance did not show an upward trend. It may be that it is difficult for the complex model structure to play the role of high-level semantic information. The accuracy of the method proposed in this paper was the highest, reaching 99.99%. At the same time, comparing different networks' macro and micro averages showed an ideal value.

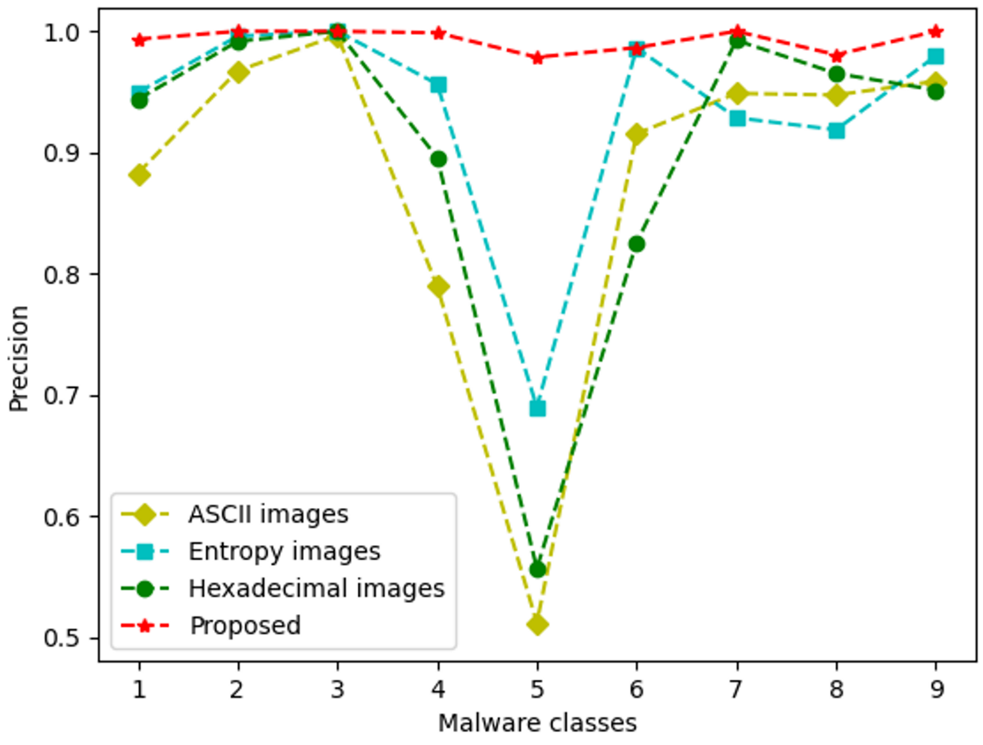

Figure 8. Precision of different grayscale images and multi-channel images.

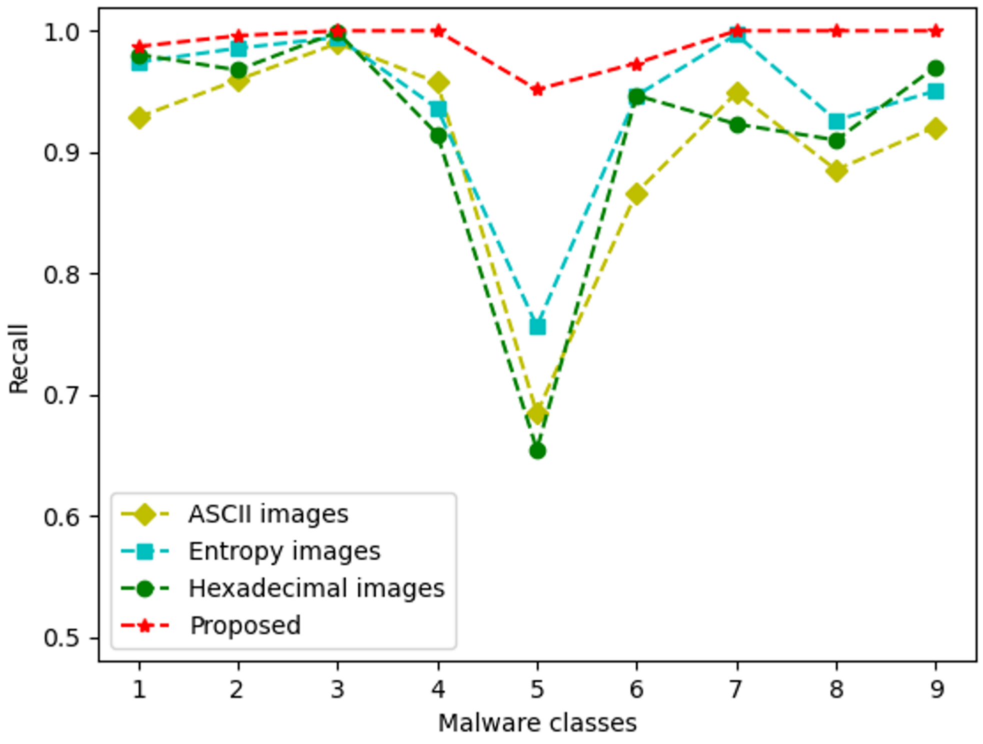

Figure 9. Recall of different grayscale images and multi-channel images.

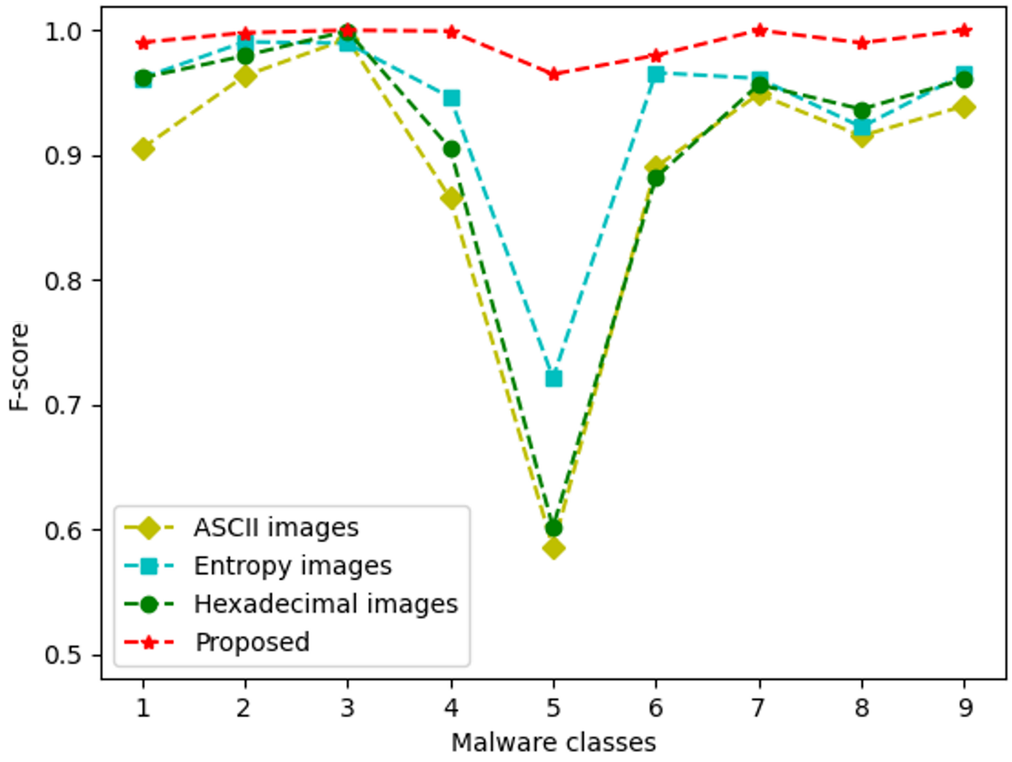

Figure 10. F-score of different grayscale images and multi-channel images.

Table 5. Performance comparison of different ResNet neural networks.

	Accuracy	Average Type	Precision	Recall	F-Score
Proposed	0.9999	micro	0.9933	0.9937	0.9935
		macro	0.9935	0.9935	0.9935
ResNet18	0.9897	micro	0.9837	0.9768	0.9801
		macro	0.9824	0.9824	0.9824
ResNet34	0.9912	micro	0.9653	0.9719	0.9685
		macro	0.9738	0.9821	0.9779
ResNet50	0.9874	micro	0.9547	0.9671	0.9608
		macro	0.9612	0.9612	0.9612
ResNet101	0.9741	micro	0.9555	0.9602	0.9578
		macro	0.9531	0.9508	0.9519
ResNet50_32x4d	0.9841	micro	0.9607	0.9587	0.9596
		macro	0.9528	0.9536	0.9531
ResNet101_32x8d	0.9695	micro	0.9571	0.9529	0.9545
		macro	0.9526	0.9571	0.9548

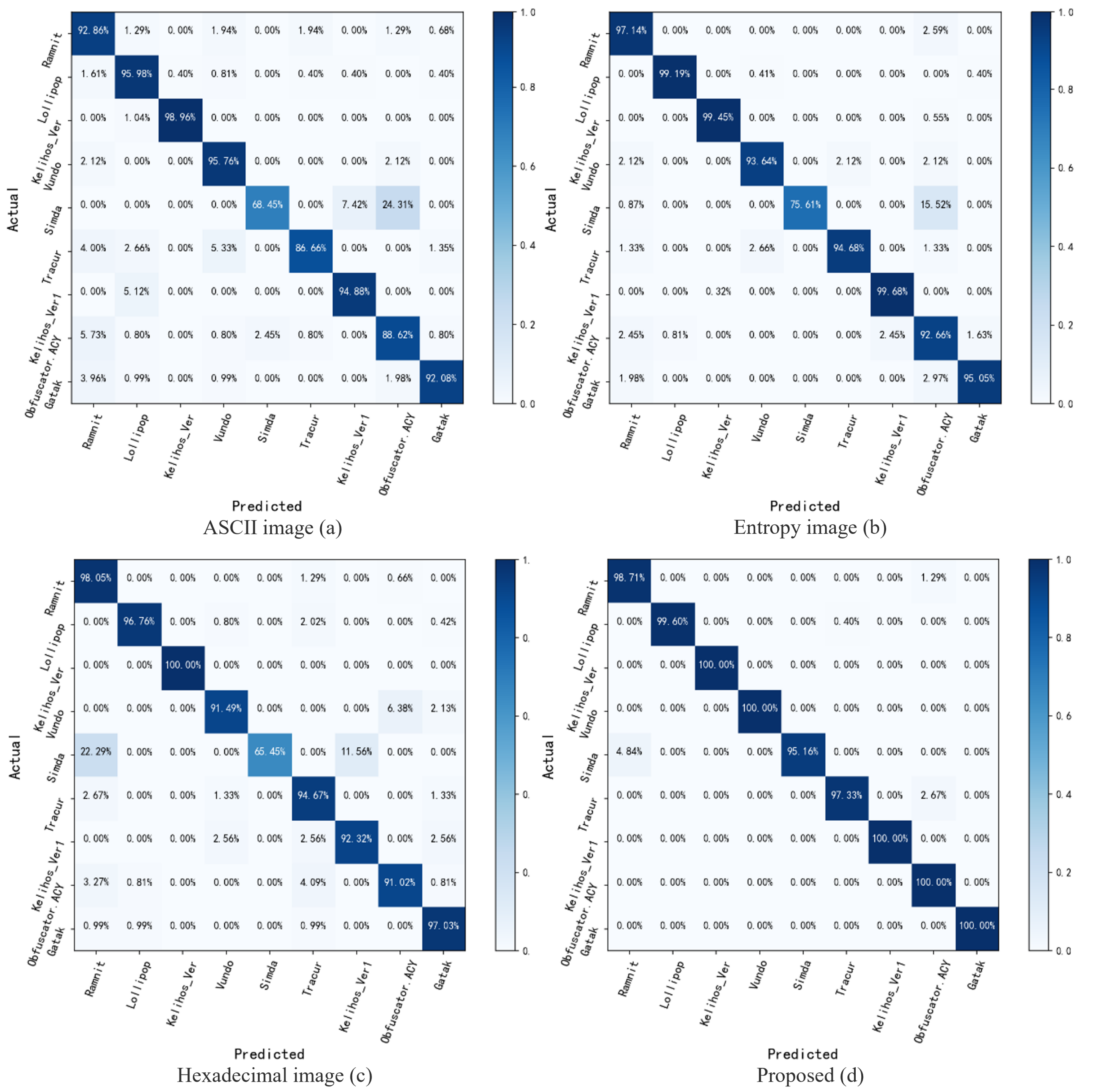

Figure 11. Confusion matrix.

4.4. Comparison with Other Methods

We compared the algorithm performance of this experiment with the performance of state-of-the-art malware classification methods in other existing literature to better evaluate the methods proposed in this experiment. By comparing the accuracy and F-score, we compare the classification results of the Microsoft dataset using different methods, as shown in Table 6, for example malware classification based on traditional methods [44–46] and malware classification based on transfer learning [47]. Gibert et al. [44] proposed a document-agnostic end-to-end deep learning system consisting of two key components: a denoising autoencoder and a dilated residual network. Alaeiyan et al. [45] proposed a system based on an integrated multi-label fuzzy classification method deployed at the edge layer. The method uses static analysis methods to extract opcode frequencies as the feature space to detect and classify malware samples. Zhu et al. [46] extracted byte and opcode features to classify malware samples based on a homology determination method using the fusion of global structure features and fine-grained local features for malware visualization. Kumar et al. [47] used a convolutional neural network based on transfer learning to classify grayscale images and introduced the early stopping technology on this basis to monitor the verification loss of the configuration parameters. It can be observed that our method achieved 99.99% on the accuracy and 99.35% on the F-score, which were superior to the existing classification technology. This showed that the algorithm can better classify malware samples and retain more effective information.

Table 6. Comparison with other relevant literature when using the Microsoft dataset.

	Method	**Accuracy**	**F-Score**
Gibert et al. [44]	Denoising autoencoders + dilated residual networks	0.9894	0.9813
Alaeiyan et al. [45]	Multi-label fuzzy clustering	0.9756	0.8921
Zhu et al. [46]	Opcode + bytecode	0.9905	0.9852
Kumar et al. [47]	CNN + transfer learning + early stopping	0.9319	\
This paper	Proposed method	0.9999	0.9935

5. Conclusions and Prospects

This paper presented a new method of malware detection. First, three different methods were used to generate grayscale images, and an interpolation algorithm was used to deal with the problem of image scale. The generated grayscale images were fused into multi-channel images. Then, we pre-trained the parameters of the model on the ImageNet dataset and combined this with the ResNet34 convolutional neural network. Finally, the improved model was used to classify the images. This method does not require feature engineering and reverse analysis and can directly extract features for training. The model uses the data enhancement method to solve the overfitting problem caused by sample imbalance.

In order to better evaluate the performance of the method proposed in this paper, the existing ResNet series of networks was extensively evaluated, and the grayscale image and color graphics were also compared and analyzed. The experimental results showed that the malware detection method proposed in this paper achieved 99.99% classification accuracy and a 99.35% F-score. The designed model has strong generalization and excellent classification ability, effectively improving malware classification accuracy.

In future work, we will classify and detect malware in reality and study a more diversified and larger collection of malware family samples. To ensure that the data results are more convincing and to avoid random errors, the next experiment will use the cross-validation method to evaluate the average results quantitatively. When processing the size of grayscale images, it is easy to cause the loss of information. Later, we hope to input images of any size into the model.

Author Contributions: Conceptualization, S.Y. and D.Z.; methodology, Z.Z.; software, Z.Z.; validation, S.Y.; formal analysis, S.Y.; writing—original draft preparation, Z.Z.; writing—review and editing, D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key Research and Development Project of China (2020AAA0107700), the National Natural Science Foundation of China (62172244), the Taishan Scholars Program (tsqn202211210), the Shandong Provincial Natural Science Foundation (ZR2020YQ06, ZR2021MF132), and the Young innovation team of colleges and universities in Shandong province (2021KJ001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Shabtai, A.; Moskovitch, R.; Elovici, Y.; Glezer, C. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *Inf. Secur. Tech. Rep.* **2009**, *14*, 16–29. [CrossRef]
2. David, B.; Filiol, E.; Gallienne, K. Structural analysis of binary executable headers for malware detection optimization. *J. Comput. Virol. Hacking Tech.* **2017**, *13*, 87–93. [CrossRef]
3. Yuxin, D.; Siyi, Z. Malware detection based on deep learning algorithm. *Neural Comput. Appl.* **2019**, *31*, 461–472. [CrossRef]
4. Liu, Y.S.; Lai, Y.K.; Wang, Z.H.; Yan, H.B. A new learning approach to malware classification using discriminative feature extraction. *IEEE Access* **2019**, *7*, 13015–13023. [CrossRef]
5. Darabian, H.; Dehghantanha, A.; Hashemi, S.; Homayoun, S.; Choo, K.K.R. An opcode-based technique for polymorphic Internet of Things malware detection. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5173. [CrossRef]
6. San, C.C.; Thwin, M.M.S.; Htun, N.L. Malicious software family classification using machine learning multi-class classifiers. In *Computational Science and Technology*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 423–433.
7. Xiao, F.; Lin, Z.; Sun, Y.; Ma, Y. Malware detection based on deep learning of behavior graphs. *Math. Probl. Eng.* **2019**, *2019*, 8195395. [CrossRef]
8. Ficco, M. Comparing API call sequence algorithms for malware detection. In Proceedings of the Workshops of the International Conference on Advanced Information Networking and Applications, Caserta, Italy, 15–17 April 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 847–856.
9. Xu, Z.; Fang, X.; Yang, G. Malbert: A novel pre-training method for malware detection. *Comput. Secur.* **2021**, *111*, 102458. [CrossRef]
10. Jian, Y.; Kuang, H.; Ren, C.; Ma, Z.; Wang, H. A novel framework for image-based malware detection with a deep neural network. *Comput. Secur.* **2021**, *109*, 102400. [CrossRef]
11. Tekerek, A.; Yapici, M.M. A novel malware classification and augmentation model based on convolutional neural network. *Comput. Secur.* **2022**, *112*, 102515. [CrossRef]
12. Kancherla, K.; Mukkamala, S. Image visualization based malware detection. In Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), Singapore, 16–19 April 2013; pp. 40–44.
13. Kancherla, K.; Donahue, J.; Mukkamala, S. Packer identification using Byte plot and Markov plot. *J. Comput. Virol. Hacking Tech.* **2016**, *12*, 101–111. [CrossRef]
14. Rezende, E.; Ruppert, G.; Carvalho, T.; Theophilo, A.; Ramos, F.; Geus, P.d. Malicious software classification using VGG16 deep neural network's bottleneck features. In *Information Technology-New Generations*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 51–59.
15. Zhao, Y.; Xu, C.; Bo, B.; Feng, Y. Maldeep: A deep learning classification framework against malware variants based on texture visualization. *Secur. Commun. Netw.* **2019**, *2019*, 4895984. [CrossRef]
16. Ren, Z.; Chen, G.; Lu, W. Malware visualization methods based on deep convolution neural networks. *Multimed. Tools Appl.* **2020**, *79*, 10975–10993. [CrossRef]
17. 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]
18. Qiao, Y.; Jiang, Q.; Jiang, Z.; Gu, L. A multi-channel visualization method for malware classification based on deep learning. In Proceedings of the 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science And Engineering (TrustCom/BigDataSE), Rotorua, New Zealand, 5–8 August 2019; pp. 757–762.
19. 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]
20. Narayanan, B.N.; Davuluru, V.S.P. Ensemble malware classification system using deep neural networks. *Electronics* **2020**, *9*, 721. [CrossRef]
21. Yuan, B.; Wang, J.; Liu, D.; Guo, W.; Wu, P.; Bao, X. Byte-level malware classification based on markov images and deep learning. *Comput. Secur.* **2020**, *92*, 101740. [CrossRef]
22. Pinhero, A.; Anupama, M.; Vinod, P.; Visaggio, C.A.; Aneesh, N.; Abhijith, S.; AnanthaKrishnan, S. Malware detection employed by visualization and deep neural network. *Comput. Secur.* **2021**, *105*, 102247. [CrossRef]
23. Yadav, P.; Menon, N.; Ravi, V.; Vishvanathan, S.; Pham, T.D. EfficientNet convolutional neural networks-based Android malware detection. *Comput. Secur.* **2022**, *115*, 102622. [CrossRef]
24. Ding, Y.; Dai, W.; Yan, S.; Zhang, Y. Control flow-based opcode behavior analysis for malware detection. *Comput. Secur.* **2014**, *44*, 65–74. [CrossRef]
25. Shalaginov, A.; Banin, S.; Dehghantanha, A.; Franke, K. *Machine Learning Aided Static Malware Analysis: A Survey and Tutorial*; Cyber Threat Intelligence; Springer: Cham, Switzerland, 2018; pp. 7–45
26. Gibert, D.; Mateu, C.; Planes, J. HYDRA: A multimodal deep learning framework for malware classification. *Comput. Secur.* **2020**, *95*, 101873. [CrossRef]

27. Wu, X.W.; Wang, Y.; Fang, Y.; Jia, P. Embedding vector generation based on function call graph for effective malware detection and classification. *Neural Comput. Appl.* **2022**, *34*, 8643–8656. [CrossRef]
28. Kakisim, A.G.; Gulmez, S.; Sogukpinar, I. Sequential opcode embedding-based malware detection method. *Comput. Electr. Eng.* **2022**, *98*, 107703. [CrossRef]
29. Bonfante, G.; Kaczmarek, M.; Marion, J.Y. Architecture of a morphological malware detector. *J. Comput. Virol.* **2009**, *5*, 263–270. [CrossRef]
30. Christodorescu, M.; Jha, S.; Seshia, S.A.; Song, D.; Bryant, R.E. Semantics-aware malware detection. In Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P'05), Oakland, CA, USA, 8–11 May 2005; pp. 32–46.
31. Bruschi, D.; Martignoni, L.; Monga, M. Detecting self-mutating malware using control-flow graph matching. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Berlin, Germany, 13–14 July 2006; Springer: Berlin/Heidelberg, Germany, 2006, pp. 129–143.
32. Lin, C.H.; Pao, H.K.; Liao, J.W. Efficient dynamic malware analysis using virtual time control mechanics. *Comput. Secur.* **2018**, *73*, 359–373. [CrossRef]
33. Sun, Y.; Bashir, A.K.; Tariq, U.; Xiao, F. Effective malware detection scheme based on classified behavior graph in IIoT. *Ad Hoc Netw.* **2021**, *120*, 102558. [CrossRef]
34. Amer, E.; Zelinka, I.; El-Sappagh, S. A multi-perspective malware detection approach through behavioral fusion of api call sequence. *Comput. Secur.* **2021**, *110*, 102449. [CrossRef]
35. Li, C.; Cheng, Z.; Zhu, H.; Wang, L.; Lv, Q.; Wang, Y.; Li, N.; Sun, D. DMalNet: Dynamic malware analysis based on API feature engineering and graph learning. *Comput. Secur.* **2022**, *122*, 102872. [CrossRef]
36. Nataraj, L.; Jacob, G.; Manjunath, B. *Detecting Packed Executables Based on Raw Binary Data*; Technical Report; University of California: Santa Barbara, CA, USA, 2010.
37. Liu, X.; Lin, Y.; Li, H.; Zhang, J. A novel method for malware detection on ML-based visualization technique. *Comput. Secur.* **2020**, *89*, 101682. [CrossRef]
38. Ni, S.; Qian, Q.; Zhang, R. Malware identification using visualization images and deep learning. *Comput. Secur.* **2018**, *77*, 871–885. [CrossRef]
39. Zhao, Z.; Zhao, D.; Li, S.; Yang, S. Malware classification based on visualization and feature fusion. In Proceedings of the 2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC), Shenzhen, China, 9–11 October 2021; pp. 53–60.
40. Conti, M.; Khandhar, S.; Vinod, P. A few-shot malware classification approach for unknown family recognition using malware feature visualization. *Comput. Secur.* **2022**, *122*, 102887. [CrossRef]
41. 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]
42. Chaganti, R.; Ravi, V.; Pham, T.D. Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification. *J. Inf. Secur. Appl.* **2022**, *69*, 103306. [CrossRef]
43. Fu, J.; Xue, J.; Wang, Y.; Liu, Z.; Shan, C. Malware visualization for fine-grained classification. *IEEE Access* **2018**, *6*, 14510–14523. [CrossRef]
44. Gibert, D.; Mateu, C.; Planes, J. An end-to-end deep learning architecture for classification of malware's binary content. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, 4–7 October 2018, Proceedings, Part III 27*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 383–391.
45. Alaeiyan, M.; Dehghantanha, A.; Dargahi, T.; Conti, M.; Parsa, S. A multilabel fuzzy relevance clustering system for malware attack attribution in the edge layer of cyber-physical networks. *ACM Trans. Cyber-Phys. Syst.* **2020**, *4*, 1–22. [CrossRef]
46. Zhu, X.; Huang, J.; Wang, B.; Qi, C. Malware homology determination using visualized images and feature fusion. *PeerJ Comput. Sci.* **2021**, *7*, e494. [CrossRef] [PubMed]
47. Kumar, S.; Janet, B. DTMIC: Deep transfer learning for malware image classification. *J. Inf. Secur. Appl.* **2022**, *64*, 103063. [CrossRef]