sensors

MDPI

Article

A Malicious Code Detection Method Based on FF-MICNN in the Internet of Things

Wenbo Zhang [1], Yongxin Feng [1], Guangjie Han [2,*], Hongbo Zhu [1] and Xiaobo Tan [1]

[1] School of Information Science and Engineering, Shenyang Ligong University, Shenyang 110159, China
[2] Department of Information and Communication Systems, Hohai University, Changzhou 213022, China
* Correspondence: hanguangjie@gmail.com

Abstract: It is critical to detect malicious code for the security of the Internet of Things (IoT). Therefore, this work proposes a malicious code detection algorithm based on the novel feature fusion–malware image convolutional neural network (FF-MICNN). This method combines a feature fusion algorithm with deep learning. First, the malicious code is transformed into grayscale image features by image technology, after which the opcode sequence features of the malicious code are extracted by the n-gram technique, and the global and local features are fused by feature fusion technology. The fused features are input into FF-MICNN for training, and an appropriate classifier is selected for detection. The results of experiments show that the proposed algorithm exhibits improvements in its detection speed, the comprehensiveness of features, and accuracy as compared with other algorithms. The accuracy rate of the proposed algorithm is also 0.2% better than that of a detection algorithm based on a single feature.

Keywords: IoT; malicious code detection; classification detection of images; improved convolutional neural network; FF-MICNN

Citation: Zhang, W.; Feng, Y.; Han, G.; Zhu, H.; Tan, X. A Malicious Code Detection Method Based on FF-MICNN in the Internet of Things. *Sensors* **2022**, *22*, 8739. https://doi.org/10.3390/s22228739

Academic Editors: Carles Gomez and Yuh-Shyan Chen

Received: 5 September 2022
Accepted: 10 November 2022
Published: 12 November 2022

1. Introduction

With the rapid development of scientific and technological information, many new core technologies have been applied in various fields. In particular, the mutual integration and innovation of artificial intelligence (AI), 5G, and the Internet of Things (IoT) promote the continuous expansion of the scale of the IoT industry, which not only has a positive influence on the industrial field, but also a significant impact on daily life [1].

In the IoT environment, sensor devices are connected through the internet protocol and they exchange information and communicate through the medium of information transmission. Due to the communication characteristics of IoT, the number of attacks, such as distributed denial of service (DDoS), buffer overflow, botnet, malicious code encryption, and ARP spoofing attacks is rapidly expanding [2]. In addition, the development of IoT technology has led to increasing demand for IoT devices. To meet this demand, many manufacturers produce devices with low security and sensitivity to vulnerabilities, thus making these devices targets of attack and further aggravating the possibility of malicious attacks. According to research by IBM, the number of internet-connected devices will increase to 50 billion by 2020. The Kaspersky Lab collected 121,588 IoT malicious code samples in 2018, about four times more than the 32,614 samples it collected in 2017. Among these samples, more than 120,000 samples of mutated IoT malicious code were found, and the attack methods were intelligently evolved [3]. To prevent IoT devices from being attacked by new or transformable malicious code, it is meaningful to detect malicious code [4].

In view of these factors, our motivation is to overcome the shortcomings of existing algorithms for malicious code detection. Aiming at the static and multi-layer features of malicious code detection, we would use the method of image and feature fusion to design

the detection algorithm, so as to improve the detection speed, the comprehensiveness of features, and the accuracy. Therefore, a malicious code detection algorithm based on FF-MICNN is proposed in the present study. The main contributions of this paper are as follows:

- A grayscale image converted from malicious code is used as the input for the improved network model, and the malicious code detection task is converted into an image classification task;
- The FF-MICNN algorithm is proposed. The opcode sequence features and grayscale image features are fused, and the improved CNN is used for detection.

The remainder of this article is organized as follows. Section 2 describes the previous work related to malicious code detection methods. Section 3 introduces the proposed FF-MICNN algorithm in detail. The simulation results and performance analysis are discussed in Section 4. Finally, Section 5 presents the conclusion and describes directions for future research.

2. Related Work

Many researchers have developed different algorithms and solutions to address the research questions discussed in the introduction. The existing methods of malicious code detection mainly include static detection algorithms [5–7] and dynamic detection algorithms [8,9].

The static detection method is to identify malicious code by analyzing the data at the data level and capturing the relevant semantic and grammatical information of the data without running the malicious code. Many scholars have carried out numerous studies on different data structures and forms as shown in Table 1.

Table 1. The static detection methods.

Authors	Algorithm Description	Merits
Zhu et al. [10]	A malicious code detection method using API sequence of malicious code.	Can quickly detect malicious code, cannot detect newly emerged malicious code.
Zhang et al. [11]	Use attribute similarity to detect malicious code.	Can quickly analyze malicious code, cannot accurately analyze confused code.
Abhijit et al. [12]	A malicious code detection method based on frequency characteristics.	Has high accuracy and can detect wrong and missed code, consumes a large number of resources.
Kang et al. [13]	An n-gram opcode features-based approach that utilizes machine learning to identify and categorize Android malware.	Supports automatic feature discovery without relying on prior experts or domain knowledge.
Imran et al. [14]	A detection scheme based on the hidden Markov model and discriminant classifier.	Relies heavily on the API sequence, and requires a large amount of computation.
Siddiquiet et al. [15]	A worm prevention technology using a data mining framework.	Can improve the detection rate of new worms without using a large amount of data.
Moser et al. [16]	A binary obfuscation scheme.	Easily avoided if malicious code is packaged or confused.

Due to the deficiency of static detection methods, dynamic detection methods have been developed as shown in Table 2. This type of detection method generates behavior reports for portable executable files by analyzing the execution code in a virtual environment and is based on the execution tracking of the code.

Table 2. The dynamic detection methods.

Authors	Algorithm Description	Merits
Hisham et al. [17]	A behavior-based feature model can dynamically analyze and evaluate a dataset of malicious code.	Can quickly detect malicious code, cannot detect newly emerged malicious code.
Li et al. [18]	A detection method based on the semantic dynamic characteristics of malicious code.	Malicious code can be detected, evasive malicious code cannot.
Rong et al. [19]	A MACSPD detection method, which is an API sequence model mining method.	Better than a similar type of algorithm in detecting unknown malicious code.
Ucci et al. [20]	A scalable clustering method to identify and group malware samples.	Can identify and group similar malware programs with better accuracy.
Phodeet et al. [21]	A model that predicts malware in executing files.	Determine the presence of malware before it executes a payload.
Mohaisenet et al. [22]	A behavior-based automated malware analysis and classification System.	Difficult to realize this method in the case of too much malicious code data.

Image detection methods are different from traditional detection methods. This type of method is improved on the basis of static detection methods via a relatively new approach to analyze the binary files of malware, and it also detects confused malware, new malware, and malware variants by converting the malware into image features. The concept of malicious code visualization has been proposed and is widely used, and realizes malicious code detection by converting malicious code into a grayscale image and further processing the image [23]. The typical image detection methods is shown in Table 3.

Table 3. The Image detection methods.

Authors	Algorithm Description	Merits
Wan et al. [24]	A malicious code classification method based on analytic behavior.	Can improve the performance of the algorithm while reducing data dimensions.
Han et al. [25]	A malicious code detection method based on the texture features of malicious code.	Cannot sufficiently solve the artificial influence, nor can it achieve end-to-end detection.
Hashemi et al. [26]	An image-based method to detect unknown malicious code.	Can be regarded as a framework with flexibility.
Xiao et al. [27]	A strategy to select a deep learning model that fits the malware visualization images.	Classification accuracy was 99.72% higher than that of other classification methods.

Based on the preceding literature review, the present work proposes a malicious code detection method based on the FF-MICNN to overcome the deficiency of the traditional detection of malicious code varieties and the use of single features. The proposed method combines both feature fusion and deep learning algorithms. First, malicious code is transformed into grayscale image features by visualization technology. Then, the opcode sequence features of the malicious code are extracted by the n-gram technique. Finally, the global and local features are fused by feature fusion technology, the obtained fusion features are input into FF-MICNN for training, and the appropriate classifier is selected for detection.

3. FF-MICNN Algorithm

With the rapid development of the IoT, many devices are connected to the Internet. Deep learning [28–31] has been applied to many fields, and its effects and achievements have attracted attention. Among the various deep learning frameworks, the CNN is the

most outstanding in image processing as compared with other networks. The algorithm proposed in this paper is a network model algorithm adapted to the malicious code environment based on the improvement of the CNN.

Figure 1 demonstrates the overall composition of the malicious code detection method based on FF-MICNN, which mainly includes three parts: data processing, model training, and classification detection. The data processing component mainly includes two processes, namely feature extraction and feature fusion. The purpose of feature extraction is to obtain the opcode sequence features and grayscale image features in the dataset with a .asm file suffix. In the feature fusion component, two extracted single features are fused to form a new feature vector, which is used as the input of the deep learning model and the training model. During model training, the parameters of the model are set and optimized by constructing the FF-MICNN, and a set of parameter combinations suitable for the training of the network model is selected so that the network can be self-adapted for classification detection. The classification detection component uses classifiers to classify malicious codes into their corresponding categories.

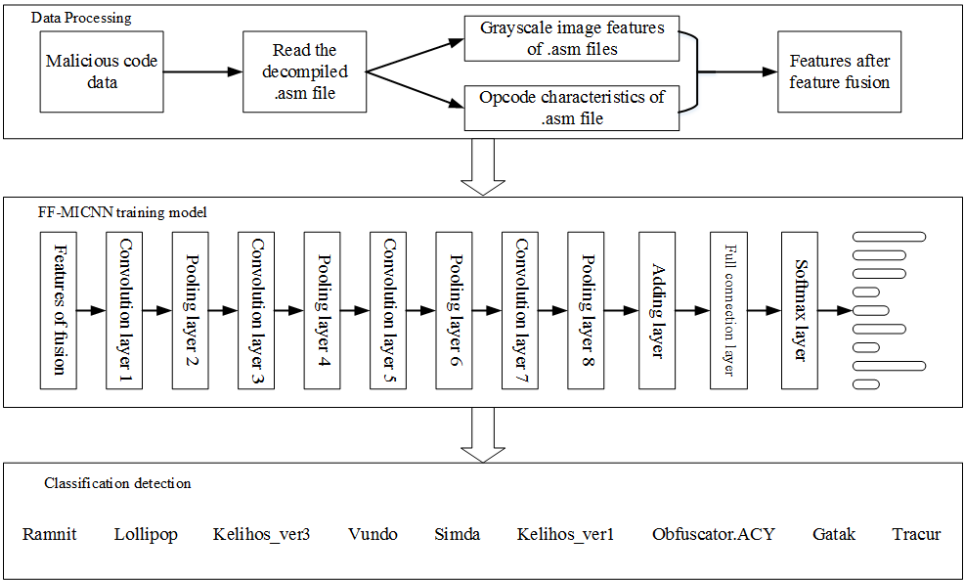

Figure 1. Malicious code detection method based on FF-MICNN.

3.1. Data Processing

3.1.1. Grayscale Image Feature Extraction

The malicious code visualization method adopts image processing technology to convert malicious code files into corresponding grayscale images, which can be used to further analyze malicious code. There is no need to execute the code to convert the malicious code into a grayscale image. Compared with the manual analysis of confused malicious code, this not only saves time and reduces the analysis of the code, but also avoids the harm caused to computers by malicious code. The conversion process of this method is shown in Figure 2, and the specific conversion process is as follows.

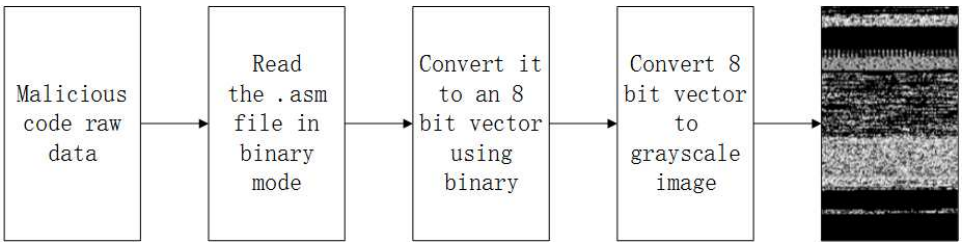

Figure 2. Malicious code visualization process.

First, the decompiled .asm files in the dataset are read in binary form in units of 8 bits. Then, the binary sequence of each unit is converted into an unsigned decimal numeric form. The decimal range is between 0 and 255, and different values represent any pixel value in the image; 0 is black, 255 is white, and other values are in between black and white. Finally, the resulting decimal value is converted into a two-dimensional array, and the process is repeated until the binary file is fully read.

3.1.2. Feature Extraction of Opcode Sequences

The opcode is located in the .text code section of the malicious code file, which describes the relevant instruction behavior of the malicious code. It can most accurately describe the local characteristics of the malicious code. The realization of the extraction of opcode sequence features is as follows.

First, the opcodes are stored in the .text code section in the .asm file, and the contents of this section must be read as the line reads. The read content is then converted in hexadecimal fashion. Next, the regular expression is used to match what is read on each line, which either contains a complete operating instruction or an instruction containing opcodes and operations. Finally, the opcodes are extracted from the matched instructions until all the opcodes of the file are extracted and the opcode sequence of the file is obtained. This algorithm is presented in Algorithm 1.

Algorithm 1 Extraction algorithm of opcode sequences

Input: .asm file;
Output: Opcode sequence;

1: Defines a variable to store an opcode sequence
2: Regularization matching
3: Open the .asm file
4: Read the file by line
5: Judgment, start reading with the .text section
6: Read the content
7: Judgment, match to the content
8: Match, extract opcode; Match failure, this content is followed by an end opcode identifier
9: Return the resulting opcode sequence
10: Finish

During feature extraction, if the extracted opcode sequence is too long, the model training cannot achieve the ideal effect. Therefore, the n-gram algorithm is used to display the opcode sequence in the form of a feature vector space. The basic idea is to manipulate the content information in the text according to the size of the byte n-value by using the sliding window method. The sequence of byte fragments with the same length of n is realized, and the frequency of the occurrence of all obtained grams with length n is counted. Then, filtering is carried out according to the set threshold, the grams that do not meet the threshold are deleted, and the gram list is formed as a reference [32]. After testing, it was found that the detection effect was the best when the value of n was 3 and the occurrence frequency was 700. Therefore, in the experiment conducted in this study, the opcode sequence features of $n = 3$ and occurrence frequency = 700 were selected.

3.1.3. Feature Fusion

At the local and global levels, the opcode sequence and grayscale image features of malicious code represent the similarity of malicious code [33]. There will be some disadvantages if the similarity of malicious code is represented only by local opcodes [34]. Although it cannot completely describe the similarity of malicious code, it can represent the features of the core code segment to some extent. The grayscale image can represent not only the features of the core code segment of malicious code but also the features of other data resource segments; however, it cannot obtain enough local feature information [35].

During the training process of the deep learning network, regardless of whether local or global features are used, there may be incomplete feature expression. Therefore, in this study, the local features of opcodes are combined with the global features of grayscale images, based on which a high-performance malicious code classification and detection method is achieved. The core algorithm of the feature fusion method used in this study is shown in Algorithm 2.

Algorithm 2 Feature fusion algorithm

Input: Opcode sequence feature $f1$ and grayscale image feature $f2$;
Output: Feature f after feature fusion;

1: Read the first opcode sequence feature $f1$
2: Read the second grayscale image feature $f2$
3: Read a label file with malicious code labels
4: Find the two features according to the ID of the malicious code
5: Superimpose the opcode sequence feature vectors on the end of the feature vectors of the grayscale image and fuse them by the function called pandas.merge()
6: Find the corresponding tag based on the ID of the malicious code and merge the two tags with the function called panda.merge()
7: Obtain the tagged feature f after feature fusion
8: Finish

3.2. Model Building

Compared with machine learning and backpropagation (BP) methods, the advantage of the CNN is the reduction of the number of parameters of the network training process via local connection and weight sharing [35,36]. However, a CNN requires that the input image size be fixed. To solve this problem, the FF-MICNN is proposed for the feature extraction and detection of malicious codes. It cannot only extract features by self-learning but also reduces the model parameters and amount of computation while ensuring accuracy [37]. This model can realize the feature extraction and detection of images of different sizes, which is an improvement based on the CNN, and its composition is demonstrated in Figure 3.

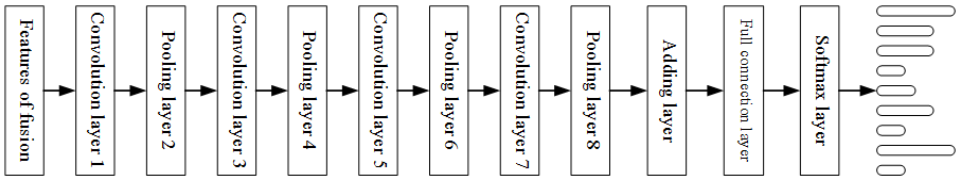

Figure 3. Network structure of malicious code detection based on FF-MICNN.

First, the input of the input layer is the fused features. Then, the convolution layer extracts the input features. The weight-sharing function of the network cannot only reduce the network parameters and retain the main features of the grayscale image but can also reduce the influence of noise. Then, the pooling layer is located behind the convolution layer. It selects the features of the feature graph output by the convolution layer, filters out the irrelevant information, realizes the dimensionality reduction of the data, and carries out multiple convolutional pooling processes to obtain the most effective features. Next, the added layer can realize the output of a fixed feature number by selecting different pooling windows. The fully connected layer can then incorporate local features that increase the output of the layer. Finally, the integration results are input into the softmax layer for category judgment to realize the detection of malicious code. The detailed iteration formula for each layer is as follows.

3.2.1. Convolution Layer

The convolution layer is the first layer in which input data are processed. The main function is to extract the characteristics of the input grayscale image. The weight-sharing function of the network can reduce the network parameters, retain the main features of the grayscale image, and reduce the influence of noise. Each neuron in the convolution layer is connected with the coefficient of the convolution value output by the upper layer, and the operation calculation is as follows:

$$x_j^l = f(\sum_{i \in M_j} x_j^{l-1} * k_{ij}^l + b_j^l), \tag{1}$$

where M_j is the input feature mapping set, k_{ij}^l is the connection of the core of the i-th input property and the weight of the j-th output feature map, and b_j^l is the offset corresponding to the j-th feature mapping.

3.2.2. Pooling Layer

The pooling layer processes the output of the convolution layer. The main function is to carry out feature selection based on the output of the convolutional layer, which cannot only filter out irrelevant information and realize the dimensionality reduction of data, but can also reduce the influence on image deformation and the dimension of image features, thus improving the accuracy of the model [38]. The operation calculation of the pooling layer is

$$x_j^l = f(down(x_j^{l-1}) + b_j^l), \tag{2}$$

where $down(.)$ represents a sub-sampling function and b_j^l is the deviation.

3.2.3. Added Layer

This layer is located before the fully connected layer and after the last pooling layer. First, the input criteria for the CNN are fixed. Second, the disassembly file of malicious code is caused by the different sizes of information storage, and the converted grayscale images are also different sizes. Hence, this cannot meet the input criteria of the network model. However, the input criteria for the network model are determined by the fully connected layer. Neurons in the fully connected layer are fixed and are fully connected to the neurons in the previous layer. Therefore, as long as the sizes of the grayscale image features are guaranteed before the fully connected layer, the standardization of the image input can be achieved. This layer is shown in Figure 4, and the specific implementation steps are as follows:

(1) The output of the convolutional layer is pooled several times, and the output of the pooling layer is improved;

(2) The normalized processing of the feature image is carried out after pooling;

(3) The obtained three feature images are cascaded;

(4) A feature image of the same size is obtained after the output.

3.2.4. Fully Connected Layer

Each neuron in the fully connected layer is fully connected to the neuron in the previous layer. The fully connected layer can integrate local features in the convolutional layer or pooling layer, as given by the following:

$$x^l = \sum \alpha^{l-1} * W^l + b^l, \tag{3}$$

where α represents the output of the previous layer, W represents the weight, and b represents the offset.

3.3. Classification

The softmax layer is selected as the classifier and final structural level for the FF-MICNN to realize the function of malicious code detection based on the extracted features. The neurons in the softmax layer are completely connected with the neurons in the upper layer. The activation function is the softmax function, which maps the result in the interval from 0 to 1. The mapped value is the probability of each category, and the probability of all categories adds up to 1.

The vector dimensions of the output of the softmax layer are determined by the number of types of malicious code datasets. In this detection model, the detection code results for which category. Therefore, the output of the softmax layer in this study is a nine-dimensional vector, which is given by the following equation:

$$f_\theta(x) = \frac{1}{e^{\theta_0 x + \theta_1 x + \cdots + \theta_j x}} \begin{bmatrix} \theta_0 x \\ \cdots \\ \theta_j x \end{bmatrix} = \begin{bmatrix} p(y=0|x,\theta) \\ \cdots \\ p(y=j|x,\theta) \end{bmatrix}, \quad (4)$$

where θ represents the parameter matrix of the neural network, $p(y|x,\theta)$ represents the probability of category Y, namely the detection result, and the category with the largest value is taken as the target category.

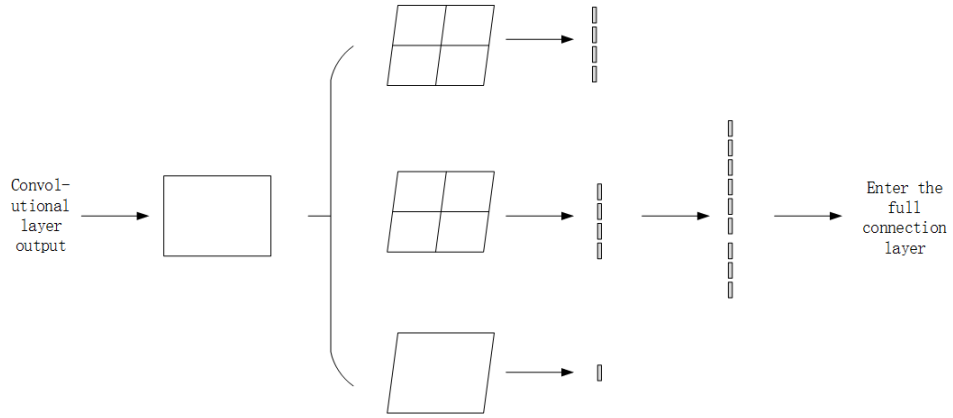

Figure 4. Process of adding layers.

4. Simulation and Analysis

4.1. Experimental Data

The data used in the Malicious Code Classification Competition initiated by Microsoft in 2015 were used in this study. These data contain nine different types of malicious code, including 10,867 samples that have been labeled. Table 4 presents the types and amounts of malicious code.

4.2. Evaluation Index

Table 5 presents the confusion matrix of the sample; TP, TN, FP, and FN can be calculated from the confusion matrix. TP represents the number of samples of a certain type of malicious code that is correctly classified as this type of malicious code after classification, TN represents the number of samples of other types of malicious codes that are correctly classified as other types of malicious codes after classification, FP represents the number of samples of certain types of malicious codes that are mistakenly classified as other types of malicious codes after classification, and FN represents the number of samples of other types of malicious codes that are mistakenly classified as these types of malicious codes after classification.

To facilitate and quantitatively analyze the detection effect of malicious code, unified evaluation indexes, including the accuracy rate, precision rate, recall rate, and F1 score, were used to evaluate the relevant performance of the model during the experiments

conducted in this study. The calculation formulas for these four evaluation indexes are, respectively, as follows.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$precision = \frac{TP}{TP + TN} \tag{6}$$

$$recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 = \frac{2 * recall * precision}{recall + precision} \tag{8}$$

4.3. Parameter Setting

In the malicious code classification method based on FF-MICNN, FF-MICNN is used to automatically extract the deep features of malicious code. To show the comprehensive features of different malicious codes and improve the malicious code classification ability, the network structure of FF-MICNN was optimized via continuous experiments and parameter adjustment during the experiments. The parameters adjusted in the experiment included the learning rate and iteration number, among others. Only a single parameter was adjusted when all other parameters remained the same. The parameter with the best generalization power was selected and set as a fixed parameter, after which the next parameter was adjusted. This was continued until all parameters were adjusted and optimized, and an optimal parameter set of FF-MICNN was obtained. The model parameter setting is shown in Table 6.

4.4. Experimental Simulation

A deep learning framework was selected as the detection model in this study. After the improvement of the model, the limitation of converting malicious code files of different sizes into two-dimensional images is solved. The input criteria for the network model are determined by the fully connected layer. The neurons in the fully connected layer are fixed and are fully connected to the neurons in the previous layer. In this study, a layer was added before the fully connected layer to ensure the size of grayscale image features. Therefore, the improved network is referred to as the FF-MICNN. The following demonstrates the effectiveness of the network model by comparing the simulation results of single and fused features.

Table 4. The types of malicious code.

Serial Number	Malicious Code Category	Description	Sample Size
1	Lollipo	—	2478
2	Ramni	Contains the code for a powerful botnet	1542
3	Simda	Consists of four types of malicious code, The most sophisticated of which are botnets, Trojans, backdoors, and password theft	42
4	Vundo	Trojans and worms	474
5	Tracur	Trojans	751
6	Gatak	Trojan horse	1013
7	Kelihos_ver3	Encrypted P2P botnets	2942
8	Kelihos_ver1	Bot	398
9	Obfuscator.ACY	Malicious code formed by a combination of four methods	1228

Table 5. The confusion matrix.

Types	Real Samples	Pseudo-Samples
Real samples	TP	FN
Pseudo-samples	FP	TN

Table 6. The parameter setting of the model network.

Parameter	Numerical Value	Instruction
Input size	variable	Fusion features of malicious code
Convolution kernel	3 × 3	Size of the convolution kernel
Step length	1	Step size of the convolution kernel and pooling window
Pool size	2 × 2	Size of the pool window
Learning rate	0.001	Learning rate of FF-MICNN
Iterations	15	Number of iterations
Activation function	ReLU	Activation function selected by FF-MICNN
Classifier	softmax	Softmax regression classification model
Dataset partition	7:3	Ratio of training data to test data

4.4.1. Simulation Experiment of Opcode Sequence

As presented in Figure 5, the traditional machine learning algorithm, deep confidence network algorithm, and CNN algorithm were selected for analysis and comparison with the proposed FF-MICNN algorithm. The figure reveals that the detection ability of each model varied under different n-gram frequencies. However, from the overall view of the figure, the detection ability of the proposed model was relatively better than those of the other models. The detection ability of the proposed model was the best when the n-gram frequency of the opcode was about 700 and was about 0.2 percentage points higher than those of the other three models. Therefore, the opcode sequence detection of the deep learning model was effective.

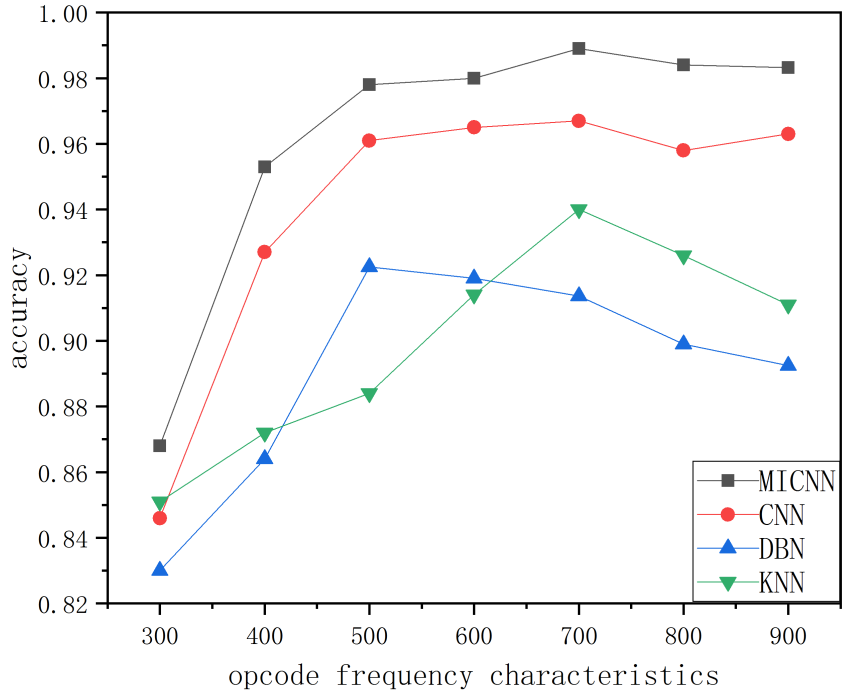

Figure 5. Simulation of opcode sequence.

4.4.2. Grayscale Image Simulation Experiment

Figure 6 presents the simulation results of the effect of the use of grayscale image features on the FF-MICNN network model and other network models. It can be seen from the graph that the proposed network model was more stable than the other models, and its accuracy tended to be stable. On average, the accuracy of the proposed network model was higher than those of the other network models, and it was also more stable.

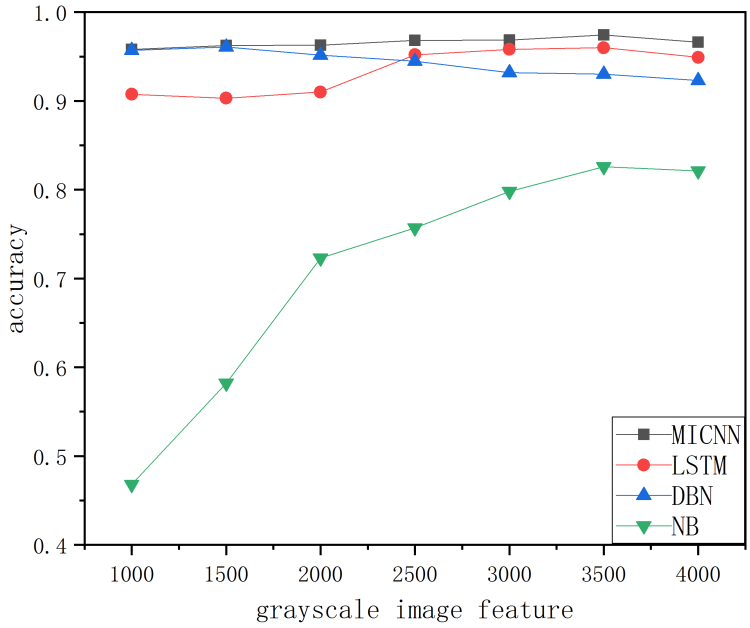

Figure 6. Simulation of gray image.

4.4.3. Simulation Detection of Fused Features

To compare the influence of fused features and single features on model classification detection, opcode sequence features, grayscale image features, and fused features were respectively detected and compared to display the detection ability of single and fused features. The comparison is shown in Figure 7.

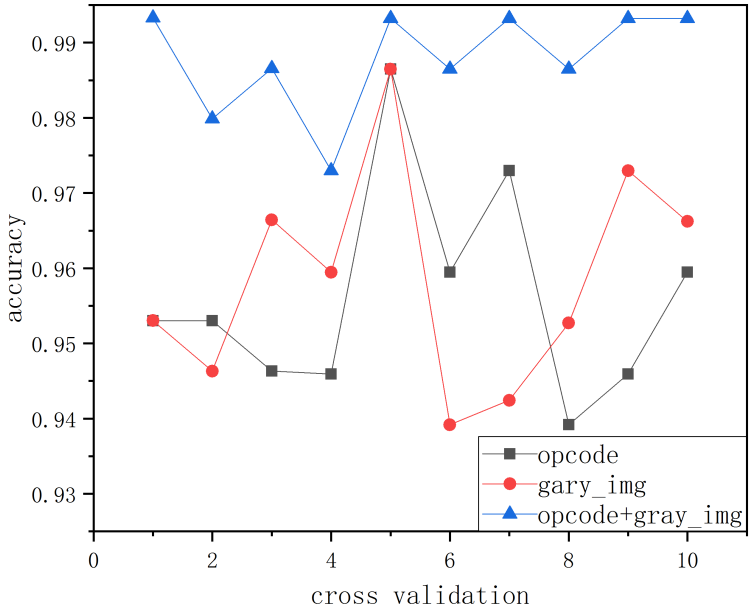

Figure 7. Simulation of single feature and fusion feature.

It can be seen from Figure 7 that the detection curve of fused features was always above the detection curve of a single feature. This indicates that feature fusion-based detection was better than single feature-based detection, and the accuracy reached 99.36%; this is 0.3% higher than the average accuracy of single feature-based detection, which is enough to demonstrate the superior detection ability of feature fusion.

The FF-MICNN was compared with the VGG15 network improved by VGG16 and a simple refinement of the CNN, as shown in Figures 8 and 9. Although the performance of the VGG15 and CNN networks was better than that of the proposed network in approximately the first 15 rounds, with the increase in the number of training rounds, the FF-MICNN outperformed them. The reason for this is that a layer was added to the proposed method to solve the problem of unequal image sizes, and the appropriate number of convolution layers and the appropriate convolution kernel size were chosen. Regarding the other two networks, one is a simple 5-layer neural network, and the other lacks the processing of the image size. From the perspective of the network structure, the proposed network is more complex; thus, more specific image features can be extracted, thereby achieving a better detection effect.

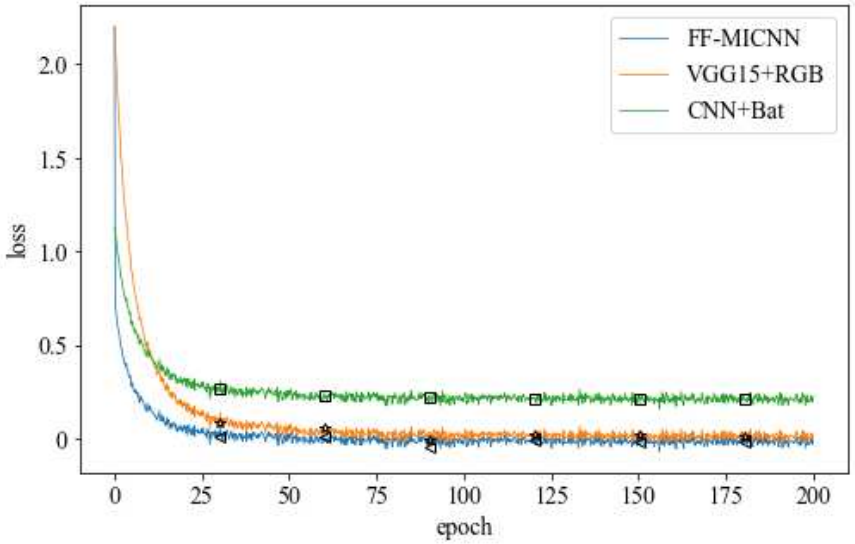

Figure 8. Change of loss in network training.

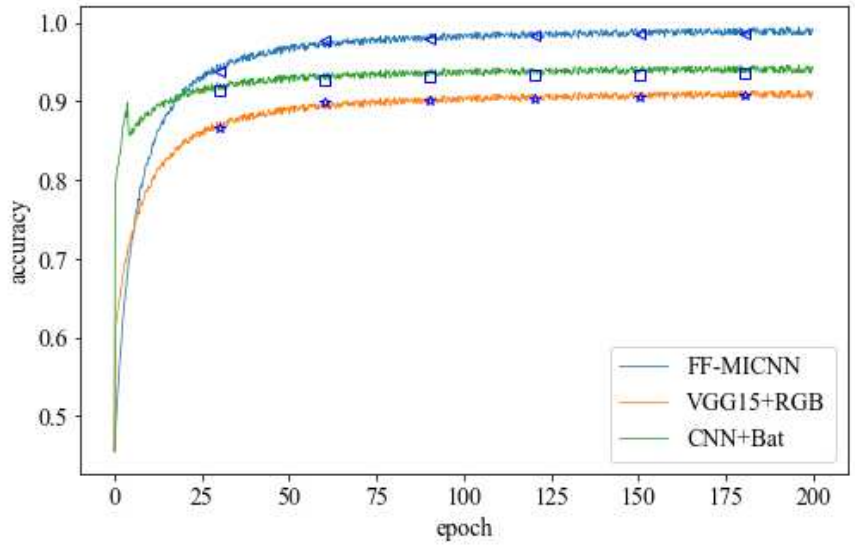

Figure 9. Change of accuracy in network training.

To verify the detection capability of the model, three network algorithms, namely a parallel CNN, a CNN, and a CNN with a balanced dataset, were selected for comparison with the proposed FF-MICNN algorithm. The model was verified to have better performance in malicious code detection. The experimental results are reported in Table 7.

Table 7. The comparison of different methods.

Algorithm Name	TPR	FPR	Accuracy	F1
Parallel-CNN [30]	0.959	0.0326	0.9818	0.9805
CNN-Image [31]	0.9387	0.0735	0.9406	0.9317
CNN+Image+Bat [32]	0.9255	0.0311	0.9346	0.9221
FF-MICNN	0.9614	0.0311	0.986	0.9817

In Table 7, the four indexes of the true positive rate (TPR), false positive rate (FPR), accuracy, and F1 value are respectively compared and analyzed. This experiment resulted in good results on the performance ability of the proposed algorithm based on the four indexes. Among them, the accuracy index can best reflect the detection ability of the proposed model; the accuracy reached 98.6%, which is better than those of the other three detection algorithms. In terms of the TPR and FPR indexes, although the comparison of the accuracy index was not obvious, the results also indicate that the detection ability of this model was the best. Regarding the F1 score, the prediction accuracy of the proposed method was also good, indicating that the method has certain advantages in the aspect of malicious code feature extraction, thus leading to better detection.

In addition, the champion detection method of the Kaggle Competition and four methods presented in the extant literature [39–42] were also selected for comparative analysis to verify the effectiveness of the proposed feature selection method. The detection algorithm of the champion team uses the image features of malicious code, the fused features of opcode and headers, and the random forest algorithm to realize code detection. In the method by Lang et al. [39], three groups of features are extracted by texture maps and disassembly files for fusion classification, and the random forest algorithm is used as the classifier for code detection. In the method by Liu et al. [40], the frequency and behavior of the opcode sequence are used as feature vectors for dynamic and static feature fusion, and the KNN algorithm is then used to detect malicious code. In the method by Li et al. [41], grayscale image and color feature vectors are fused, and malicious code detection is then realized via the random forest algorithm. In the method by Luo et al. [42], texture image features and command frequency features are used for feature fusion, and a deep confidence network is used for detection. These five algorithms were compared with the proposed FF-MICNN algorithm, and the results show that the FF-MICNN model achieved better performance in malicious code detection. The experimental results are presented in Table 8.

Table 8. Comparison of different methods.

Paper Author	Accuracy	Description
Kaggle champion	98%	Feature fusion of image features, opcodes, and headers
Lang et al.	87%	Feature fusion of opcode sequences and gist features
Xiu et al.	96%	Feature fusion of opcode sequence frequency and behavior
Li et al.	95.518%	Feature fusion of grayscale image and color
Luo et al.	95.76%	Feature fusion of grayscale image texture and operation frequency
FF-MICNN	98.6%	Truncated scale grayscale image and opcode sequence

According to the analysis method described previously, two characteristic features were selected for use in the proposed method, and very high detection accuracy was achieved. It was verified that the two features selected for use in the proposed method

can sufficiently express the global and local features of malicious code, and almost all the features of malicious code can be extracted for detection. In addition, the proposed method can automatically learn and extract deep-level features; this is different from the machine learning algorithm, which extracts only surface-level features.

The fusion of the two features extracted from the FF-MICNN model can achieve a comprehensive representation of malicious code features, which can improve the classification detection of the model to a certain extent. In addition, the fused features can reveal more superficial features than single features in the representation of malicious code features, and deeper features can be further obtained through the FF-MICNN model, thus improving the results of malicious code detection. Therefore, the proposed model has a certain significance for the detection of malicious samples. It not only achieves detection with less time and resource consumption but also is not subject to the amount and type of malicious code, and can solve the problems of the explosive growth of malicious code and analysis difficulty.

In this work, a malicious code detection method based on the combination of the sequence features of the opcodes and grayscale image features was proposed. This algorithm mainly extracts the features of the sequence of operation codes as the local features and extracts the features of malicious code disassembly files converted into grayscale images as the global features. The two features are fused through a fusion algorithm and input into the FF-MICNN for comprehensive abstract feature extraction. Classifiers are used to divide them into respective categories to complete the detection of malicious code. The experimental results show that feature fusion is more conducive to the representation of code features and can represent more comprehensive and abstract features, which aids in the detection of malicious code.

5. Conclusions

This paper proposed a malicious code detection algorithm to solve the problems of the constant increase of malicious code and deformation based on malicious code. First, the confused code causes the expression of extracted features to be unclear, and its behavior and instructions are difficult to understand. Therefore, the algorithm can avoid the confusion of malicious code by converting the code into grayscale image space vectors, and the detection of malicious code can be converted into an image detection task. Second, the grayscale image input into the CNN limits the size of the grayscale image, so the extracted feature expression is incomplete. Hence, the CNN is improved by adding a structural layer to realize the function of the input of grayscale images without limiting the size, thereby improving the detection performance. The results of simulations demonstrated that this algorithm is better than the machine learning algorithm in terms of the objective indicators of the accuracy, recall rate, and precision rate, and more accurate detection results were obtained.

Aiming at the problem of the incomplete feature description of malicious code, a feature fusion detection algorithm was proposed. First, the opcode sequence in the disassembled .asm file of malicious code is selected as the local feature, and the .asm file is then transformed into a grayscale image space vector as the global feature. The static features of the malicious code can be described completely by fusing these two features. The fused features are input into the FF-MICNN to realize automatic feature extraction, and the types of malicious code are represented by the softmax layer in the form of probability, thus realizing the detection of malicious code. The results of simulations revealed that the proposed algorithm can fully express the characteristics of malicious code because of the fusion of local and global features. The proposed FF-MICNN realizes the deep feature extraction and autonomous learning of malicious code and avoids human interference. The results of the simulations showed that the accuracy of the proposed algorithm was better than those of algorithms using a single feature.

Although the proposed algorithm is optimized for the graphical processing of malicious code, it is characterized by some shortcomings. In the present study, the test dataset

was unbalanced and the accuracy was meaningful in the specified condition. Furthermore, the test dataset size was not large, which cannot indicate the superiority of FF-MICNN. Moreover, the algorithm does not take into account the uneven distribution of malicious code types in the dataset. In future research, the test dataset will be balanced and enlarged to further improve the proposed malicious code detection algorithm under these conditions.

Author Contributions: W.Z. researched the extant literature, conceived the study concepts, designed the improved convolutional neural network, proposed the malicious code detection algorithm based on FF-MICNN, carried out the simulation, analyzed the simulation results, and took charge of the entire manuscript. Y.F. and G.H. assisted with the integrity of the entire study, provided crucial intellectual support, and revised the manuscript. H.Z. transformed malicious code into grayscale image features, extracted the opcode sequence features of malicious code, and fused the global and local features. X.T. contributed to polishing the revised version of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: The work is supported by the National Key Research and Development Program under grant no. 2017YFE0125300, China Academy of Military Sciences Fund (2019), Liaoning Distinguished Professor Project (2017), the National Natural Science Foundation of China–Guangdong Joint Fund under grant no. U1801264, the Jiangsu Key Research and Development Program under grant no. BE2019648, Project of Shenzhen Science and Technology Innovation Committee under grant no. JCYJ20190809145407809, National Science and Technology Major Project under grant no. 2017-V-0011-0062, the project of Fujian University of Technology under grant no. GY-Z19066.

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

References

1. Shen, G.; Chen, Z.; Wang, H.; Chen, H.; Wang, S. Feature fusion-based malicious code detection with dual attention mechanism and BiLSTM. *Comput. Secur.* **2022**, *119*, 1–14. [CrossRef]
2. Trivikram, M.; Nir, N. Improving malicious email detection through novel designated deep-learning architectures utilizing entire email. *Neural Netw.* **2022**, *in press*.
3. Wang, Q.; Qian, Q. Malicious code classification based on opcode sequences and textCNN network. *J. Inf. Secur. Appl.* **2022**, *67*, 1–12. [CrossRef]
4. Hou, J.; Liu, F.; Lu, H.; Tan, Z.; Zhuang, X.; Tian, Z. A novel flow-vector generation approach for malicious traffic detection. *J. Parallel Distrib. Comput.* **2022**, *169*, 72–86. [CrossRef]
5. Malka, N. Estimation of the success probability of a malicious attacker on blockchain-based edge network. *Comput. Netw.* **2022**, *in press*.
6. RAsim, M.; Fargana, J.; SAbira, S. Image-based malicious Internet content filtering method for child protection. *J. Inf. Secur. Appl.* **2022**, *65*, 103123.
7. Lara, K.; Divakaran, L. Predicting stock market returns from malicious attacks: A comparative analysis of vector autoregression and time-delayed neural networks. *Decis. Support Syst.* **2022**, *51*, 745–759.
8. Marcus, B.; Marco, Z.; Daniela, O.; Andre, G. HEAVEN: A Hardware-Enhanced AntiVirus ENgine to accelerate real-time, signature-based malware detection. *Expert Syst. Appl.* **2022**, *201*, 117083.
9. Wu, J.; Wang, W.; Huang, L.; Zhang, F. Intrusion detection technique based on flow aggregation and latent semantic analysis. *Appl. Soft Comput.* **2022**, *127*, 109375. [CrossRef]
10. Zhu, J.; Wu, Z.; Guan, Z. API Sequences Based Malware Detection for Android. In Proceedings of the Ubiquitous Intelligence & Computing & IEEE Intl Conf on Autonomic & Trusted Computing & IEEE Intl Conf on Scalable Computing & Communications & Its Associated Workshops, Beijing, China, 21 July 2016; pp. 673–676.
11. Zhang, F.; Zhao, T. Malware Detection and Classification Based on N-Grams Attribute Similarity. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; pp. 793–796.
12. Abhijit, Y.; Maninder S. Malware detection based on opcode frequency. In Proceedings of the 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), Pyeongchang, South Korea, 31 January–3 February 2016; pp. 646–649.
13. Kang, B.; Yerima, S.Y.; Sezer, S.; McLaughlin, K. N-gram Opcode Analysis for Android Malware Detection. *Int. J. Cyber Situat. Aware.* **2016**, *1*, 231–254. [CrossRef]
14. Imran, M.; Afzal, M.T.; Qadir, M.A. Similarity-Based Malware Classification Using Hidden Markov Model. In Proceedings of the 2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), Jakarta, Indonesia, 29–31 October 2015; pp. 129–134.
15. Siddiquiet, M.; Wang, M.; Lee, J. Detecting Internet Worms Using Data Mining Techniques. *J. Syst. Cybern. Inform.* **2008**, *6*, 48–53.

16. Moser, A.; Kruegel, C.; Kirda, E. Limits of Static Analysis for Malware Detection. In Proceedings of the Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), Miami Beach, FL, USA, 10–14 December 2007; pp. 421–430.
17. Hisham, S.G.; Yousef, B.M.; Mohammed, A.A. Behavior-based features model for malware detection. *J. Comput. Virol. Hacking Tech.* **2016**, **12**, 59–67.
18. Li, M.; Jia, X.; Wang, R.; Lin, D. A Feature Selection and Modelling Method for Malicious Code. *Comput. Appl. Softw.* **2015**, *32*, 266–271.
19. Rong, F.; Zuo, Z.; Fang, Y. MACSPMD: Malicious Code Detection Based on Malicious API Call Sequence Pattern Mining. *Comput. Sci.* **2018**, *45*, 131–138.
20. Ucci, D.; Aniello, L.; Baldoni, R. Survey of machine learning techniques for malware analysis. *Comput. Secur.* **2019**, *81*, 123–147. [CrossRef]
21. Davuluru, V.S.P.; Narayanan, B.N.; Balster, E.J. Convolutional Neural Networks as Classification Tools and Feature Extractors for Distinguishing Malware Programs. In Proceedings of the 2019 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 15–19 July 2019; pp. 273–278.
22. Mohaisen, A.; Alrawi, O.; Mohaisen, M. AMAL: High-fidelity, behavior-based automated malware analysis and classification. *Comput. Secur.* **2015**, *52*, 251–266. [CrossRef]
23. Liu, Y.; Wang, Z.; Hou, Y. Malware visualization and automatic classification with enhanced information density. *J. Tsinghua Univ.* **2019**, *59*, 9–14.
24. Wan, L.; Xia, J.; Zhu, Y.; Lv, Z. An Improved Semi-supervised Feature Selection Algorithm Based on Information Entropy. *Stat. Decis.* **2021**, *17*, 66–70.
25. Han, X.; Qu, W.; Yao, X.X.; Guo, C.Y.; Zhou, F. Research on Malicious Code Variant Detection Method Based on Texture Fingerprint. *J. Commun.* **2014**, *35*, 125–136.
26. Hashem, H.; Ali, H. Visual malware detection using local malicious pattern. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 1–14. [CrossRef]
27. Xiao, G.; Li, J.; Chen, Y.; Li, K. MalFCS: An effective malware classification framework with automated feature extraction based on deep convolutional neural networks. *J. Parallel Distrib. Comput.* **2020**, *141*. [CrossRef]
28. Chu, Q.; Liu, G.; Zhu, X. Visualization Feature and CNN Based Homology Classification of Malicious Code. *Chin. J. Electron.* **2020**, *29*, 154–160. [CrossRef]
29. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
30. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
31. Ke, J.; Lin, R.; Sharma, A. An Automatic Instrument Recognition Approach Based on Deep Convolutional Neural Network. *Recent Adv. Electr. Electron. Eng.* **2021**, *14–16*. [CrossRef]
32. Qiang, H.; Guo, Y.; Tian, L. Research on malicious code detection method based on deep belief network. *Comput. Technol. Dev.* **2019**, *29*, 93–97.
33. Kumar, R.; Zhang, X.; Wang, W.; Khan, R.U.; Kumar, J.; Sharif, A. A Multimodal Malware Detection Technique for Android IoT Devices Using Various Features. *IEEE Access* **2019**, *7*, 64411–64430. [CrossRef]
34. Ren, W.; Zhai, L.; Jia, J.; Wang, L.; Zhang, L. Learning selection channels for image steganalysis in spatial domain. *Neurocomputing* **2020**, *401*, 10012–10026. [CrossRef]
35. Chechlinski, U.; Siemitkowska, B.; Majewski, M. A System for Weeds and Crops Identification-Reaching over 10 FPS on Raspberry Pi with the Usage of MobileNets, DenseNet and Custom Modifications. *Sensors* **2019**, *19*, 3787 [CrossRef]
36. Hamzeh, A.; Bakhshinejad, N. Parallel-CNN Network for Malware Detection. *IET Inf. Secur.* **2019**, *14*, 210–219.
37. Gibert, D.; Mateu, C.; Planes, J. Using convolutional neural networks for classification of malware represented as images. *J. Comput. Virol. Hacking Tech.* **2019**, *15*, 15–28. [CrossRef]
38. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.; Chen, J. Detection of Malicious Code Variants Based on Deep Learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [CrossRef]
39. Lang, D.; Ding, W.; Jiang, H.; Chen, Z. Malicious Code Classification Algorithm Based on Multi-feature Fusion. *J. Comput. Appl.* **2019**, *39*, 2333–2338.
40. Xiu, Y.; Liu, J. Malware Detection Based on Opcode Sequence Frequency Vector and Behavior Feature Vector. *Inf. Secur. Commun. Priv.* **2016**, *9*, 97–101.
41. Li, S.; Wang, C.; Shi, Y. Malicious Code Detection Based on Multi-feature Random Forest. *Comput. Appl. Softw.* **2020**, *37*, 328–333.
42. Luo, S. Research on Deep Learning Malicious Code Analysis and Detection Technology. Ph.D. Thesis, Xinjiang University, Ürümqi, China, 2018.