applied sciences

MDPI

Article

Malware Classification Using Convolutional Fuzzy Neural Networks Based on Feature Fusion and the Taguchi Method

Cheng-Jian Lin [1,*], Min-Su Huang [1] and Chin-Ling Lee [2]

[1] Department of Computer Science and Information Engineering, National Chin-Yi University of Technology, Taichung 411, Taiwan
[2] Department of International Business, National Taichung University of Science and Technology, Taichung 404, Taiwan
* Correspondence: cjlin@ncut.edu.tw

Abstract: The applications of computer networks are increasingly extensive, and networks can be remotely controlled and monitored. Cyber hackers can exploit vulnerabilities and steal crucial data or conduct remote surveillance through malicious programs. The frequency of malware attacks is increasing, and malicious programs are constantly being updated. Therefore, more effective malware detection techniques are being developed. In this paper, a convolutional fuzzy neural network (CFNN) based on feature fusion and the Taguchi method is proposed for malware image classification; this network is referred to as FT-CFNN. Four fusion methods are proposed for the FT-CFNN, namely global max pooling fusion, global average pooling fusion, channel global max pooling fusion, and channel global average pooling fusion. Data are fed into this network architecture and then passed through two convolutional layers and two max pooling layers. The feature fusion layer is used to reduce the feature size and integrate the network information. Finally, a fuzzy neural network is used for classification. In addition, the Taguchi method is used to determine optimal parameter combinations to improve classification accuracy. This study used the Malimg dataset to evaluate the accuracy of the proposed classification method. The accuracy values exhibited by the proposed FT-CFNN, proposed CFNN, and original LeNet model in malware family classification were 98.61%, 98.13%, and 96.68%, respectively.

Keywords: malware image classification; convolutional neural network; fuzzy theory; Taguchi method; feature fusion

Citation: Lin, C.-J.; Huang, M.-S.; Lee, C.-L. Malware Classification Using Convolutional Fuzzy Neural Networks Based on Feature Fusion and the Taguchi Method. *Appl. Sci.* **2022**, *12*, 12937. https://doi.org/10.3390/app122412937

Academic Editor: Oscar Reinoso García

Received: 14 November 2022
Accepted: 14 December 2022
Published: 16 December 2022

1. Introduction

With the rapid development of information technology, the progress of network technology has made our lives more convenient. However, malware can penetrate information technology devices through loopholes in program security, which can cause system damage, limited network bandwidth, and the theft of crucial files. The frequency of malware attacks has been increasing. For example, antivirus company Kaspersky Lab detected 69,277,289 unique malicious objects in 2016 [1]. McAfee Labs reported that 670 million malware samples were detected in 2017 [2]. According to Malwarebytes' annual malware report, more than 50 million cyber threats were detected separately in 2018 and 2019 [3,4]. The 2020 Trend Micro Cybersecurity Report stated that 119,000 cyberattacks occur every minute [5]. Because of the wide variety of malware and the increasing number of malware attacks, various malware classification methods have been proposed by researchers.

Malware analysis can be broadly categorized into static analysis and dynamic analysis. Static analysis involves parsing and extracting the features of malicious programs without executing code. Common static analysis is a signature-based method that involves searching a signature database for matching signatures to determine whether the program is malware. A local signature database is required to store signatures extracted from malware by experts. The signature database must be regularly manually updated to effectively

prevent new malware attacks. However, this approach has major limitations. Researchers and malware authors have demonstrated that malware can evade signature-based detection through new program encryption or obfuscation techniques [6–9]. Chen et al. [10] proposed the use of four easily extractable small-scale features to classify malware families and used machine learning methods to search for the best model and hyperparameters for each feature and parameter combination. Kazuki et al. [11] developed a malware analysis system that conducts control traffic analysis, antisink editing, feature extraction, and similarity calculation. Peyman [12] proposed the use of heuristic-based malware detection and simultaneous byte analysis based on static features.

In dynamic analysis, malicious programs (such as virtual machines or sandboxes) are executed in a controlled environment. Malware detection results are obtained from the collection and analysis of the system operating behavior, network packets, file storage, and download items [13]. Mohaisen et al. [14] proposed a malware classification technology called the automated malware and labeling scheme (AMAL), which is based on automated behavior analysis. AMAL primarily uses AutoMal and MaLabel to monitor the file system and network behavior. MaLabel classifies similar malware through extracted behavioral monitoring. Support vector machine (SVM), decision tree, and K-nearest neighbor (KNN) algorithms are used to classify specific malware families. Galal et al. [15] proposed a dynamic analysis method based on malware behavior. In this method, application programming interface (API) hooking technology is used to determine the parameters of the malware and collect relevant information; from the extracted API features, parameters, and sequences, the unique malware behavior is inferred. The decision tree, random forest, and SVM algorithms are then used to classify malware. Kolosnjaji et al. [16] used a recurrent neural network and convolutional neural network (CNN) for feature extraction and an n-gram for malware detection. Fahade and Wei [17] proposed the use of the longest common substring and longest common subsequence for character matching to detect malware. Damodaran et al. [18] used numerous malware samples for the comparison of static and dynamic malware analysis.

The aforementioned static and dynamic analyses have many limitations. Although static analysis is fast and safe, it cannot be used to classify unknown malware. Dynamic analysis can be used for accurately identifying the code and determining the functionality of a malicious program; however, this method might affect the application of the computer. Therefore, some researchers have advocated the conversion of malware from binary files to grayscale images and then utilizing various algorithmic techniques to classify these malware images [19]. In contrast to static and dynamic analysis, malware image classification does not require strong malware domain knowledge, and it bypasses the need for malware fine-tuning to overcome obfuscation techniques. Malware image classification thus allows for fast classification applicable to various malware types [20].

Machine learning technology has been widely used to address the malware detection problem. Narayanan et al. [21] used principal component analysis to extract features, and artificial neural networks (ANN), KNN, and SVM were then used to complete the classification of malware images. Garcia and Muga II [22] used random forest to classify malware images, and they reported satisfactory classification results. Gao et al. [23] proposed a malware classification framework based on malware visualization and semisupervised learning. This framework mainly includes three parts: the parts for malware visualization, feature extraction, and classification. Feature fusion methods are used to fuse local and global features to save time and improve feature correlation. Nataraj et al. [24] proposed a classification method based on standard image features. This method is simple and efficient and requires neither disassembly nor code execution. The above-mentioned methods require the user to define the features in advance.

In recent years, many scholars have conducted in-depth research with deep learning approaches on malware detection. Lin et al. [25] used LeNet for malware family classification. They used convolution operations to automatically extract malware features. Kalash et al. [26] proposed a deep learning architecture to classify malware samples. Qi

et al. [27] used an adversarial learning framework for unsupervised domain adaptation to enable gradient boosting decision trees to learn domain-invariant features and to mitigate performance degradation in the target domain. Because CNNs have too many parameters and require high-performance hardware, Lin and Jhang [28] employed convolution operations with fuzzy neural network to reduce the number of network parameters in breast cancer classification and obtained good performance. Because CNNs require numerous parameters and these parameters are difficult to determine, the trial-and-error method is widely used for parameter selection. To reduce the time and cost of experiments, the Taguchi method [29] can be used to statistically optimize parameter selection by using an orthogonal array of influencing factors and their levels.

In this paper, a convolutional fuzzy neural network (CFNN) based on feature fusion and the Taguchi method (FT-CFNN) is proposed for malware image classification. The FT-CFNN comprises two parts: a CFNN and Taguchi-method-based optimal parameter combinations. The CFNN comprises convolutional, pooling, feature fusion, and fuzzy neural network (FNN) layers. Four feature fusion methods, namely global max pooling (GMP) fusion, global average pooling (GAP) fusion, channel global max pooling (CGMP) fusion, and channel global average pooling (CGAP) fusion, are proposed to reduce the feature size and integrate the network information. An FNN is used for classification. In addition, the Taguchi method is used to determine the affecting factors and the best parameter combination for achieving optimal accuracy. In this study, we focus on improving deep learning methods that require a large number of learning parameters and improving classification accuracy by optimizing parameter combinations. The major contributions of this study are as follows:

- An efficient FT-CFNN is proposed for malware image classification.
- Four feature fusion methods, namely GMP fusion, GAP fusion, CGMP fusion, and CGMP fusion, are proposed to reduce the feature size and integrate network information.
- The size of the adjustable parameters can be reduced by replacing a fully connected network with an FNN.
- To reduce the number of experiments required for the various parameter combinations, the Taguchi method is used to determine the affecting factors and levels.

The rest of this paper is organized as follows. Section 2 describes the structure of the proposed FT-CFNN and the Taguchi method. Section 3 presents the experimental results obtained when using the proposed FT-CFNN. Finally, Section 4 details the conclusions of this study and recommendations for future research.

2. Proposed FT-CFNN

In this section, the classification method of the proposed FT-CFNN is described in detail. Figure 1 shows a flowchart detailing the process of malware image classification with the aforementioned network. First, the Taguchi method is used to select the affecting factors and their levels. Next, the training image set is input into the FT-CFNN to implement the Taguchi experiment. The network then determines whether the experimental results meet the requirements of users. If the results do not meet the requirements of users, the affecting factor and its level are reselected. If the results meet the requirements of users, the experiment is concluded. Finally, the malware images are classified into their corresponding categories.

2.1. Convolutional Fuzzy Neural Network

The architecture of the CFNN of the FT-CFNN is displayed in Figure 2. It comprises convolution, pooling, feature fusion, and FNN layers. Two convolution layers and two pooling layers are used in this network. Four feature fusion methods, namely GMP fusion, GAP fusion, CGMP fusion, and CGMP fusion, are proposed to reduce the feature size and integrate the network information. To reduce the size of adjustable parameters, an FNN is used to replace a fully connected network.

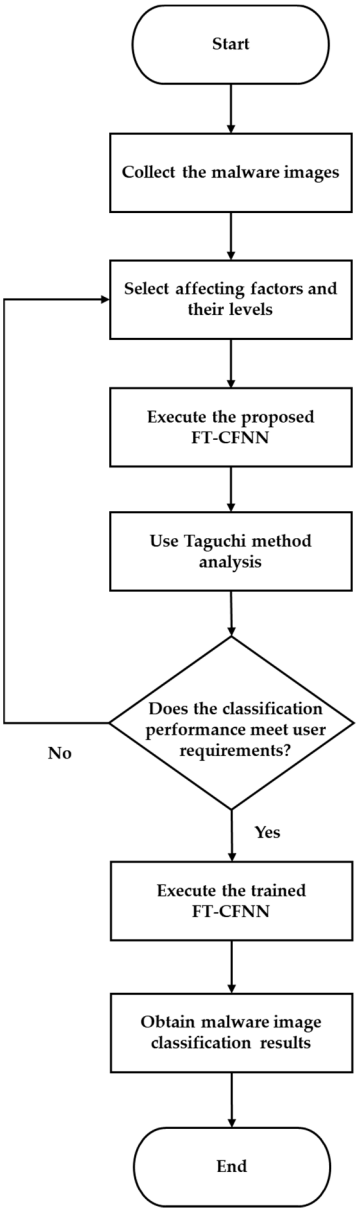

Figure 1. Flowchart of malware image classification with the FT-CFNN.

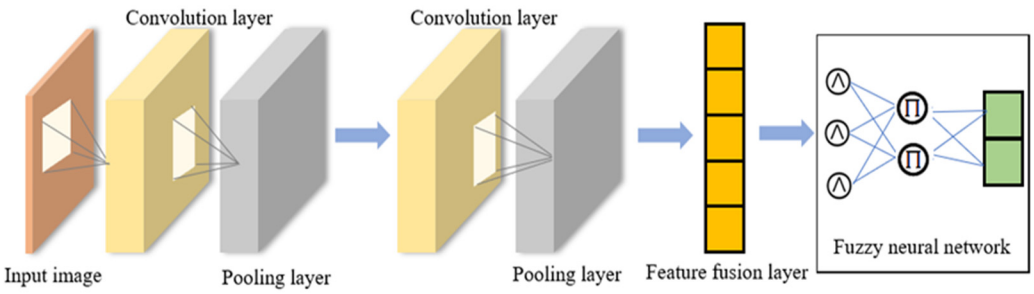

Figure 2. Architecture of the CFNN used in the FT-CFNN.

2.1.1. Convolution Layers

The convolutional layers of the CFNN contain multiple convolution kernels. A convolution kernel is used to extract the features of an image by moving it from left to right and from top to bottom through a sliding action. The inner product operation is performed on overlapping positions to obtain the feature values for these points.

2.1.2. Pooling Layers

The main function of a pooling layer is to reduce the size of input image features and retain only important features. Commonly used pooling operations are of two types: maximum pooling and average pooling. Maximum pooling is used to determine the maximum value in a convolutional kernel. Average pooling is used to compute the average value in a convolution kernel.

2.1.3. Feature Fusion Layer

In this study, four feature fusion methods (GMP fusion, GAP fusion, CGMP fusion, and CGMP fusion) were used (Figure 3). Figure 3a illustrates the global pooling fusion method. First, two feature maps are completed, and each feature map (width and height) is then fused separately. Finally, a feature point is obtained. Figure 3b illustrates channel global pooling fusion. All channels between feature maps are fused (depth) in this method. The operations used in the aforementioned methods are of two types: maximum value operations and average value operations.

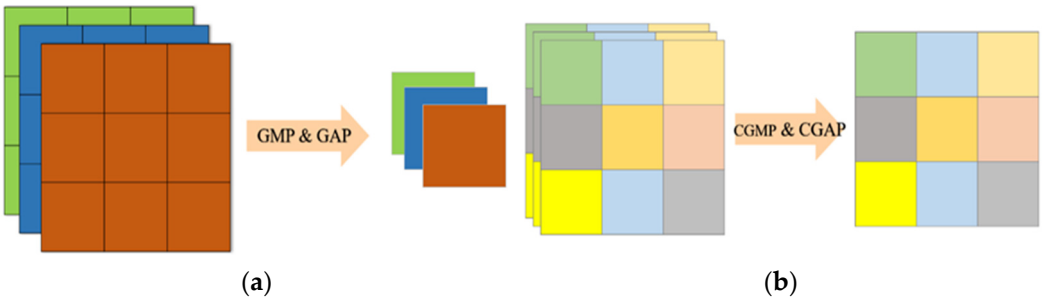

Figure 3. Feature fusion methods: (**a**) GMP fusion and GAP fusion as well as (**b**) CGMP fusion and CGAP fusion.

2.1.4. FNN Layer

An FNN combines the logical reasoning of humans and the learning abilities of neural networks. FNNs have been successfully used in classification systems [30,31]. Figure 4 illustrates the architecture of an FNN, which primarily contains a fuzzification layer, fuzzy rule layer, and defuzzification layer. Fuzzy rules are inferred using IF–THEN representations. These rules are expressed as follows:

$$Rule_j: \text{ IF } x_1 \text{ is } A_{1j} \text{ and } x_2 \text{ is } A_{2J} \ldots \text{ and } x_n \text{ is } A_{nj}, \text{THEN } y_j \text{ is } w_j$$

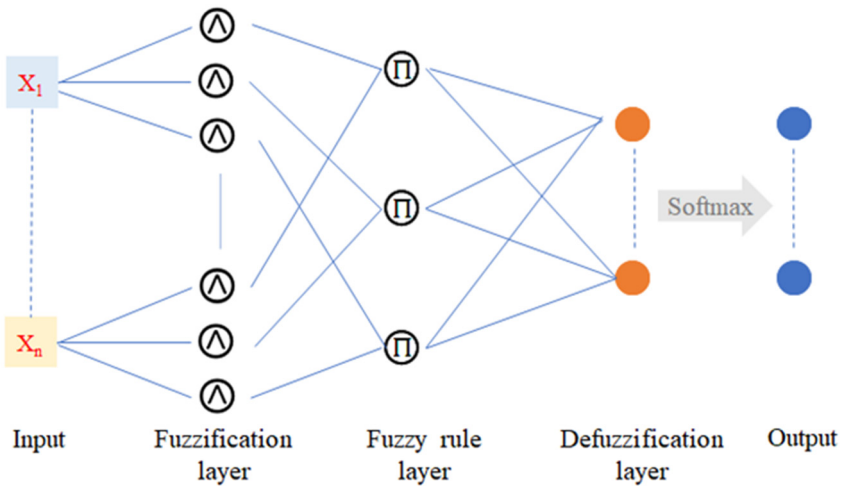

Figure 4. Architecture of an FNN.

In the aforementioned rule, x_i is the input, A_{ij} is the membership function, and w_j is the output of the jth rule.

In the fuzzification layer, the Gaussian membership function used by each input is fuzzified to obtain the corresponding membership values. The relevant formula is as follows:

$$\mu_{ij}(x) = exp\left(-\frac{\left[x_i - m_{ij}\right]^2}{\sigma_{ij}^2}\right) \tag{1}$$

where x_i is the input, m_{ij} is the mean value, and σ_{ij} is the deviation.

Subsequently, the AND operation is completed for the attribution values corresponding to each input to obtain the excitation intensity of each fuzzy rule. The most commonly used AND operations are minimum and product. To facilitate the derivation of the back-propagation algorithm, the product operation is used in this study. The excitation intensity of each fuzzy rule is calculated as follows:

$$R_j = \prod_{i=1}^{n} \mu_{ij} \tag{2}$$

where μ_{ij} is the excitation value of each membership function.

Finally, each rule is input into the defuzzification layer to obtain a crisp output. The relevant formula is as follows:

$$y_k = \sum_{j=1}^{r} R_j w_{jk} \tag{3}$$

where y_k is the kth output, R_j is the excitation intensity of the jth fuzzy rule, r is the number of fuzzy rules, and w_{jk} is the weight of the jth fuzzy rule and kth output.

2.2. Taguchi Method

The Taguchi method is a statistical method for experimental design. This method is a low-cost, high-efficiency, and high-quality engineering method. In a complete factorial experiment design, the number of experiments to be conducted increases as the number of factors increases. Therefore, an orthogonal array (OA) is developed to analyze the signal-to-noise (S/N) ratio of various parameter combinations by conducting the least number of experiments. The S/N ratio is selected as the indicator of quality optimization and is used to analyze the experimental effect of each factor and level to achieve quality optimization. In this study, the measured values constitute the experimental evaluation index. When the measured value is higher, the quality loss is lower. The relevant formula is as follows:

$$S/N = -10log\left(1/n\sum_{i=1}^{n}\frac{1}{y_i^2}\right) \tag{4}$$

Because orthogonal tables are reliable, they are widely used in several fields [32]. The Taguchi method reduces the required number of experiments to be conducted and the time required to complete an experiment, thereby reducing engineering costs and enabling a reliable impact factor to be obtained. From the various types of orthogonal tables, an appropriate orthogonal table is selected according to the number of factors and the number of levels. The notation used for orthogonal tables is $Lx(y^z)$, where L represents the abbreviation of the orthogonal table (a Latin square), x represents the number of experiments, y represents the number of levels, and z represents the maximum number of factors that can be accommodated.

3. Experimental Results

We designed experiments to evaluate the accuracy of malware image classification with the developed FT-CFNN. In this section, the source of the dataset is stated, and the selection of FT-CFNN parameters and the network settings are then described. The Taguchi method

was used to determine the optimal parameter combination. The experimental results of the proposed FT-CNN were then compared with those obtained using other methods.

3.1. Dataset

The Malimg dataset, which is a public database obtained from Virus-Share.com, was adopted in this study. Currently, samples on approximately 47 million types of viruses are available on this website. A total of 9339 images of 25 types of viruses [33] were used in this study (Table 1).

Table 1. Numbers of images of various types of viruses adopted in this study [33].

Type	Virus Species	Number of Images
1	Adialer.C	122
2	Agent.FYI	116
3	Allaple.A	2949
4	Allaple.L	1591
5	Alueron.gen!J	198
6	Autorun.K	106
7	C2LOP.gen!g	200
8	C2LOP.P	146
9	Dialplatform.B	177
10	Dontovo.A	162
11	Fakerean	381
12	Instantaccess	431
13	Lolyda.AA1	213
14	Lolyda.AA2	184
15	Lolyda.AA3	123
16	Lolyda.AT	159
17	Malex.gen!J	136
18	Obfuscator.AD	142
19	Rbot!gen	158
20	Skintrim.N	80
21	Swizzor.gen!E	128
22	Swizzor.gen!I	132
23	VB.AT	408
24	Wintrim.BX	97
25	Yuner.A	800
Total number of images		9339

3.2. Experimental Results Obtained with the CFNN

In the adopted CFNN, four fusion methods (GMP fusion, GAP fusion, CGMP fusion, and CGAP fusion) are used to perform malware image classification. This network contains two convolutional layers and two pooling layers, and an FNN is used in it to replace a fully connected network. Therefore, the CFNN has a small architecture, a small number of parameters, and a high operation speed. Table 2 presents the learning parameter settings of the CFNN, and Table 3 presents the network architecture parameter settings of the CFNN. The test loops of TensorFlow and Keras were used for deep learning in this study. The parameters of the CFNN were the same in the experiments conducted using the four

fusion methods. The adopted dataset contained 9339 malware images, 80% of which were randomly used as the training set, and the remaining 20% of the images were used as the test set. In the first convolution and pooling layers, the filter size was 32, the size of the convolution kernel was 3 × 3, and the stride was 2. In the second convolution and pooling layers, the filter size was 64, the size of the convolution kernel was 3 × 3, and the stride was 2. The feature fusion layer was used to reduce the dimensions of the feature layer. Finally, the FNN was used for classification.

Table 2. Learning parameter settings of the CFNN.

Image Size	224 × 224 × 3
Epochs	50
Learning rate	0.001
Batch size	64
Image Size	224 × 224 × 3
Epochs	50

Table 3. Network architecture parameter settings of the CFNN.

Layer	Kernel Size	Number	Layer	Kernel Size
Convolution_1	3 × 3	32	2	0
Max_pooling_1	2 × 2	-	2	
Convolution_2	3 × 3	64	2	0
Max_pooling_2	2 × 2	-	2	
Feature fusion	-	64	-	
Fuzzy Rule Layer	-	64	-	
DeFuzzify Layer	-	25	-	

Accuracy was used as the evaluation index of the proposed network. This parameter is determined as follows:

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FN}) + (\text{FP} + \text{TN})} \tag{5}$$

where TP, FP, TN, and FN represent the numbers of true positives, false positives, true negatives, and false negatives, respectively. Table 4 presents the accuracy achieved with the four fusion methods. The accuracy values achieved by the CFNN with the GMP, GAP, CGMP, and CGAP fusion methods were 96.68%, 97.86%, 98.13%, and 97.64%, respectively. The experimental results indicate that the proposed CFNN achieved the highest accuracy with the CGAP fusion method.

Table 4. Accuracy of the CFNN with various fusion methods.

Method	Fusion Methods	Accuracy
CFNN	GMP	97.86%
	GAP	96.68%
	CGMP	97.64%
	CGAP	98.13%

3.3. Experimental Results Obtained with the FT-CFNN

As presented in Table 4, the accuracy obtained with CGAP fusion was higher than that achieved with the other three fusion methods. Because the network architecture parameters

could not be easily determined, the Taguchi method was used to determine the impact factor and optimize the parameter combinations. Therefore, the developed FT-CFNN uses the CGAP fusion method to apply the Taguchi method.

In the conducted experiment, six factors were selected: Conv1-Filter (F1), Conv1-Kernel size (K1), Conv1-Padding (P1), Conv2-Filter (F2), Conv2-Kernel size (K2), and Conv2-Padding (*P*2). Four of these factors (F1, K1, F2, and K2) are three-level factors, and the other two factors (P1 and P2) are two-level factors. The affecting factors and their levels are presented in Table 5. According to the total number of factors and levels in the experiment, we used the L36 orthogonal table, which is composed of different factors and levels.

Table 5. Affecting factors and their levels.

No.	Abbreviation	Affecting Factors	Level 1	Level 2	Level 3
A	F1	Conv1-Filter	8	16	32
B	K1	Conv1-Kernel size	3 × 3	5 × 5	6 × 6
C	P1	Conv1-Padding	0	1	
D	F2	Conv2-Filter	16	32	64
E	K2	Conv2-Kernel size	3 × 3	5 × 5	6 × 6
F	P2	Conv2-Padding	0	1	

The selected three- and two-level factors required $2^2 \times 3^4 = 324$ traditional experiments. After the Taguchi method was applied, only 36 experiments were required. Therefore, the Taguchi method reduced the number of experiments required. To ensure the stability of the experiments, three experiments were performed for each parameter combination in the OA, and the average of the three accuracy values was used to calculate the S/N ratio, as presented in Table 6.

Table 6. Accuracy values and S/N ratios for different parameter combinations.

No.	F1	K1	P1	F2	K2	P2	Expt. 1 Accuracy	Expt. 2 Accuracy	Expt. 3 Accuracy	Average	S/N Ratio
1	8	3	0	16	3	0	0.9791	0.9823	0.9759	0.9791	−0.1836
2	16	5	0	32	5	0	0.9818	0.9850	0.9807	0.9813	−0.1534
3	32	6	0	64	6	0	0.9813	0.9721	0.9818	0.9784	−0.1899
4	8	3	0	32	3	0	0.9786	0.9754	0.9796	0.9779	−0.1945
5	16	5	0	64	5	0	0.9855	0.9807	0.9829	0.983	−0.1487
6	32	6	0	16	6	0	0.9818	0.9796	0.9823	0.9812	−0.1646
7	16	3	0	64	3	0	0.9802	0.9780	0.9764	0.9783	−0.1915
8	32	5	0	16	5	0	0.9778	0.9754	0.9802	0.9778	−0.1951
9	8	6	0	32	6	0	0.9767	0.9695	0.9839	0.9767	−0.2052
10	32	3	0	32	3	1	0.9636	0.9630	0.9657	0.9647	−0.3176
11	8	5	0	64	5	1	0.9711	0.9668	0.9363	0.9581	−0.3755
12	16	6	0	16	6	1	0.9689	0.9802	0.9738	0.9743	−0.2264
13	32	3	0	16	5	1	0.9716	0.9764	0.9614	0.9698	−0.2669
14	8	5	0	32	6	1	0.9813	0.9754	0.9743	0.977	−0.2022
15	16	6	0	64	3	1	0.9673	0.9689	0.9700	0.9681	−0.2759
16	32	3	0	32	5	1	0.9668	0.9673	0.9663	0.9668	−0.2933
17	8	5	0	64	6	1	0.9786	0.9839	0.9818	0.9814	−0.1628

Table 6. *Cont.*

No.	F1	K1	P1	F2	K2	P2	Expt. 1 Accuracy	Expt. 2 Accuracy	Expt. 3 Accuracy	Average	S/N Ratio
18	16	6	0	16	3	1	0.9646	0.9588	0.9689	0.9641	−0.3178
19	8	3	1	64	5	0	0.9813	0.9684	0.9759	0.9752	−0.2185
20	16	5	1	16	6	0	0.9780	0.9818	0.9754	0.9786	−0.1898
21	32	6	1	32	3	0	0.9684	0.9721	0.9770	0.9703	−0.2424
22	16	3	1	64	5	0	0.9762	0.9770	0.9754	0.9758	−0.2092
23	32	5	1	16	6	0	0.9754	0.9732	0.9764	0.975	−0.2199
24	8	6	1	32	3	0	0.9721	0.9796	0.9834	0.9784	−0.1903
25	16	3	1	16	6	0	0.9777	0.9780	0.9775	0.9776	−0.1954
26	32	5	1	32	3	0	0.9788	0.9802	0.9775	0.9788	−0.1857
27	8	6	1	64	5	0	0.9791	0.9668	0.9823	0.9761	−0.211
28	16	3	1	32	6	1	0.9561	0.9614	0.9545	0.9573	−0.3789
29	32	5	1	64	3	1	0.9649	0.9663	0.9636	0.9649	−0.3099
30	8	6	1	16	5	1	0.9689	0.9652	0.9689	0.9677	−0.2855
31	32	3	1	64	6	1	0.9644	0.9625	0.9663	0.9644	−0.3149
32	8	5	1	16	3	1	0.9614	0.9604	0.9695	0.9638	−0.3208
33	16	6	1	32	5	1	0.9759	0.9668	0.9796	0.9741	−0.2283
34	8	3	1	32	6	1	0.9534	0.9738	0.9593	0.9622	−0.336
35	16	5	1	64	3	1	0.9641	0.9679	0.9695	0.9672	−0.29
36	32	6	1	16	5	1	0.9754	0.9780	0.9480	0.9754	−0.2929

The S/N ratio for each factor and level combination was obtained from the results of the 36 experiments based on larger-the-better characteristic. Table 7 presents the difference of S/N ratio, significance ranking, optimal level, and optimal parameter combination for each factor. If the difference value of S/N ratio for each factor was higher, the corresponding factor had a stronger influence. The significance ranking represents the ranking of the influence degree of each factor. A higher difference of S/N ratio for P2 indicated that a factor had a stronger influence (Table 7). The final optimal parameter combination was as follows: F1 = 16, K1 = 5, P1 = 0, F2 = 16, K2 = 6, and P2 = 0.

Table 7. Results obtained for each factor.

Level	F1	K1	P1	F2	K2	P2
1	−0.2405	−0.2582	−0.2258	−0.2382	−0.2517	−0.1938
2	−0.2338	−0.2295	−0.2567	−0.244	−0.2399	−0.2887
3	−0.2494	−0.2359		−0.2415	−0.2322	
Difference	0.0147	0.0281	0.03	0.0057	0.0194	0.0937
Significance ranking	5	3	2	6	4	1
Best level	2	2	1	1	3	1
Optimal parameter combination	16	5	0	16	6	0

The results of analysis of variance (ANOVA) are presented in Table 8, and these results indicate whether the screened experimental factors are significant with respect to the experimental results. In Table 8, the degree of freedom is the number of levels minus 1, SS is the error sum of squares, the size of the *F* value determines the significance of the factors affecting the overall experiment, and the contribution degree represents the proportion of

each factor to the overall experimental results. Table 8 indicates that the factor P2 (i.e., the padding in second layer) has the strongest influence on and highest contribution to the experiments.

Table 8. Results of ANOVA.

Factor	Degree of Freedom	SS	F	Contribution Rate
Conv1-Filter (F1)	2	0.001479	0.43	1.5%
Conv1-Kernel size (K1)	2	0.005514	1.54	5.6%
Conv1-Padding (P1)	1	0.008557	4.81	8.7%
Conv2-Filter (F2)	2	0.000047	0.01	0%
Conv2-Kernel size (K2)	2	0.002313	0.64	2.3%
Conv2-Paddinjg (P2)	1	0.080943	45.53	81.9%
Error	25	0.044441		
Total	35	0.143294		100%

Table 9 presents the optimal parameter combinations obtained using the Taguchi method. These parameters were set for the FT-CFNN, which was then used for malware image classification. The accuracy and sensitivity values obtained with the CFNN and FT-CFNN are presented in Table 10. Experimental results show that the accuracy and sensitivity of the proposed FT-CFNN with CGAP fusion method are 98.61% and 97.96%, which are 0.48% and 0.45% higher than CFNN, respectively.

Table 9. Optimal parameter combination for the FT-CFNN.

Layer	Kernel Size	Number of Filters	Stride	Padding
Convolution_1	5×5	16	2	0
Max_pooling_1	2×2	-	2	
Convolution_2	6×6	16	2	0
Max_pooling_2	2×2	-	2	
Feature fusion	-	64	-	
Fuzzy Rule Layer	-	64	-	
DeFuzzify Layer	-	25	-	

Table 10. Accuracy and sensitivity values obtained with the CFNN and FT-CFNN.

Method	Fusion Method	Accuracy	Sensitivity
CFNN	CGAP	98.13%	97.51%
FT-CFNN		98.61%	97.96%

3.4. Comparison of the Results Obtained with Various Methods

To verify the effectiveness of the FT-CFNN, we compared its accuracy with the accuracy values reported in other studies [22–27] conducted on the Malimg dataset. In this experiment, Garcia and Muga II [22], Gao et al. [23], and Nataraj et al. [24] classified malware images using traditional machine learning methods such as random forest, KNN, and logistic regression. Although these methods require only a small number of learning parameters and obtain good malware image classification results (Table 11), the user is required to define the malware image features in advance. Lin et al. [25], Kalash et al. [26], and Qi et al. [27] used deep learning technology to obtain good malware image classification results. However, these methods require a large number of learning parameters (Table 11). The experimental results indicated that the proposed FT-CFNN is more accurate

than the methods used in the previous relevant studies [22–27] (Table 11). Furthermore, the proposed FT-CFNN requires only 13,609 parameters and achieves 98.61% accuracy.

Table 11. Accuracy values obtained by various methods and the number of parameters required by these methods.

Methods		Fusion Method	Accuracy	Parameters
Garcia and Muga II [22]			95.62%	<1000
Gao et al. [23]			97.95%	<1000
Nataraj et al. [24]			97.18%	<1000
Lin et al. [25]			97.05%	5,408,561
Kalash et al. [26]			98.52%	134,362,969
Qi et al. [27]			93.37%	555,329
Proposed method	CFNN	GAP	96.68%	29,209
		GMP	97.86%	29,209
		CGAP	98.13%	22,681
		CGMP	97.64%	22,681
	FT-CFNN	CGAP	98.61%	13,609

4. Conclusions

In this paper, an FT-CFNN is proposed for malware image classification. This network comprises two parts: a CFNN and Taguchi-method-based optimal parameter combinations. Four feature fusion methods (GMP, GAP, CGMP, and CGMP) are proposed for the CFNN to reduce the feature size and integrate network information. An FNN is used to replace a fully connected network to reduce the size of adjustable parameters. The proposed CFNN achieved accuracy values of 96.68%, 97.86%, 98.13%, and 97.64% with the GMP, GAP, CGMP, and CGMP fusion methods, respectively, for the Malimg dataset. The experimental results indicated that the CFNN achieved the highest accuracy with the CGAP fusion method. To reduce the number of experiments for parameter combinations, the Taguchi method was used to determine the affecting factors and levels. The experimental results indicated that the accuracy of the FT-CFNN, proposed CFNN, and original LeNet model in malware family classification was 98.61%, 98.13%, and 96.68%, respectively.

Because the Malimg dataset only contains 9339 images, generative adversarial networks can be used in the future to increase the number of images and thus improve stability. Indicators such as accuracy, sensitivity, and specificity have been widely used by scholars for comparing the classification and identification performance of deep learning networks. However, these parameters have different values in each network learning training step; therefore, the use of a single value or index average to assess a network's image classification performance might lead to inaccurate assessments. Therefore, future research should address this problem by using statistical methods to define a performance recognition index. Furthermore, in this study, we only use the existing malware database for analysis and classification, and do not apply the proposed method to actual cyberattack problems. Therefore, we will also apply the proposed method to practically any malicious programs that steal crucial data in future research.

Author Contributions: Conceptualization, C.-J.L. and M.-S.H.; Methodology, C.-J.L., M.-S.H. and C.-L.L.; Software, C.-J.L. and M.-S.H.; Data Curation, C.-J.L. and C.-L.L.; Writing—Original Draft Preparation, C.-J.L. and C.-L.L.; Supervision, C.-J.L.; Funding Acquisition, C.-J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology of the Republic of China, grant number MOST 110-2221-E-167-031-MY2.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

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

References

1. Kaspersky Report: Kaspersky Security Bulletin: Overall Statistics for 2016. Available online: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07182326/Kaspersky_Security_Bulletin_2016_Statistics_ENG.pdf (accessed on 2 October 2022).
2. McAfee Report: McAfee Labs Threats Report in June 2017. Available online: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2017.pdf (accessed on 15 August 2022).
3. Gao, X.; Hu, H.; Shan, C.; Liu, B.; Niu, Z.; Xie, H. Malware classification for the cloud via semi-supervised transfer learning. *J. Inf. Secur. Appl.* **2020**, *55*, 102661. [CrossRef]
4. Malwarebytes Labs. 2020 State of Malware Report. Available online: https://www.malwarebytes.com/resources/files/2020/02/2020_state-of-malware-report.pdf (accessed on 7 August 2022).
5. A Constant State of Flux: Trend Micro 2020 Annual Cybersecurity Report. Available online: https://www.trendmicro.com/zh_tw/about/newsroom/press-releases/2021/2021-03-16.html (accessed on 22 July 2022).
6. Yan, J.; Yong, Q.; Rao, Q. Detecting malware with an ensemble method based on deep neural network. *Secur. Commun. Netw.* **2018**, *2018*, 7247095. [CrossRef]
7. Moser, A.; Kruegel, C.; Kirda, E. Limits of static analysis for malware detection. In Proceedings of the IEEE Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), Miami Beach, FL, USA, 10–14 December 2007; pp. 421–430.
8. Zakeri, M.; Daneshgar, F.F.; Abbaspour, M. A static heuristic approach to detecting malware targets. *Secur. Commun. Netw.* **2015**, *8*, 3015–3027. [CrossRef]
9. Alsmadi, T.; Alqudah, N. A Survey on malware detection techniques. In Proceedings of the 2021 International Conference on Information Technology (ICIT), Amman, Jordan, 14–15 July 2021; pp. 371–376.
10. Chen, Z.; Brophy, E.; Ward, T. Malware classification using static disassembly and machine learning. *arXiv* **2021**. [CrossRef]
11. Iwamoto, K.; Wasaki, K. Malware classification based on extracted API sequences using static analysis. In Proceedings of the Asian Internet Engineeering Conference, Bangkok, Thailand, 14 November 2012; pp. 31–38.
12. Khodamoradi, P.; Fazlali, M.; Mardukhi, F.; Nosrati, M. Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms. In Proceedings of the 2015 18th CSI International Symposium on Computer Architecture and Digital Systems (CADS), Tehran, Iran, 7–8 October 2015; pp. 1–6.
13. Willems, C.; Holz, T.; Freiling, F. Toward automated dynamic malware analysis using cwsandbox. *IEEE Secur. Priv.* **2007**, *5*, 32–39. [CrossRef]
14. Mohaisen, A.; Alrawi, O.; Mohaisen, M. AMAL: High-fidelity, behavior-based automated malware analysis and classification. *Comput. Secur.* **2015**, *52*, 251–266. [CrossRef]
15. Galal, H.S.; Mahdy, Y.B.; Atiea, M.A. Behavior-based features model for malware detection. *J. Comput. Virol. Hacking Tech.* **2015**, *12*, 59–67. [CrossRef]
16. Kolosnjaji, B.; Zarras, A.; Webster, G.; Eckert, C. Deep learning for classification of malware system call sequences. In Proceedings of the 29th Australasian Joint Conference on Artificial Intelligence (AI), Hobart, Australia, 5–9 December 2016; pp. 137–149.
17. Mira, F.; Huang, W. Performance Evaluation of String Based Malware Detection Methods. In Proceedings of the 2018 24th International Conference on Automation and Computing (ICAC), Newcastle Upon Tyne, UK, 6–7 September 2018; pp. 1–6.
18. Damodaran, A.; Troia, F.D.; Visaggio, C.A.; Austin, T.H.; Stamp, M. A comparison of static dynamic and hybrid analysis for malware detection. *J. Comput. Virol. Hacking Tech.* **2015**, *13*, 1–12. [CrossRef]
19. Han, K.S.; Kang, B.J.; Im, E.G. Malware analysis using visualized image matrices. *Sci. World J.* **2014**, *1*. [CrossRef] [PubMed]
20. Azab, A.; Khasawneh, M. Msic: Malware spectrogram image classification. *IEEE Access* **2020**, *8*, 102007–102021. [CrossRef]
21. Narayanan, B.N.; Djaneye-Boundjou, O.; Kebede, T.M. Performance analysis of machine learning and pattern recognition algorithms for Malware classification. In Proceedings of the 2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS), Dayton, OH, USA, 16 February 2017; pp. 338–342.
22. Garcia, F.C.C.; Muga, F.P. Random forest for malware classification. *arXiv* **2016**. [CrossRef]
23. Gao, T.; Zhao, L.; Li, X.; Chen, W. Malware detection based on semi-supervised learning with malware visualization. *Math. Biosci. Eng.* **2021**, *18*, 5995–6011. [CrossRef] [PubMed]
24. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Article No. 4, New York, NY, USA, 20 July 2011; pp. 1–7.
25. Lin, C.J.; Lin, X.Y.; Jhang, J.Y. Malware classification using a Taguchi-based deep learning network. *Sens. Mater.* **2022**, *34*, 3569–3580. [CrossRef]

26. Kalash, M.; Rochan, M.; Mohammed, N.; Bruce, N.D.B.; Wang, Y.; Iqbal, F. Malware classification with deep convolutional neural networks. In Proceedings of the 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 26–28 February 2018; pp. 1–5.
27. Qi, P.; Wang, W.; Zhu, L.; Ng, S.K. Unsupervised domain adaptation for static malware detection based on gradient boosting trees. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM '21), Queensland, Australia, 1–5 February 2021; pp. 1457–1466.
28. Lin, C.J.; Jhang, J.Y. Intelligent traffic-monitoring system based on YOLO and convolutional fuzzy neural networks. *IEEE Access* **2022**, *10*, 14120–14133. [CrossRef]
29. Rezania, A.; Atouei, S.A.; Rosendahl, L. Critical parameters in integration of thermoelectric generators and phase change materials by numerical and Taguchi methods. *Mater. Today Energy* **2020**, *16*, 100376. [CrossRef]
30. Shalaginov, A.; Grini, L.S.; Franke, K. Understanding neuro-fuzzy on a class of multinomial malware detection problems. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 684–691.
31. Shihabudheen, K.V.; Pillai, G.N. Recent advances in neuro-fuzzy system: A survey. *Knowl. Based Syst.* **2018**, *152*, 136–162. [CrossRef]
32. Lin, C.J.; Li, Y.C. Lung nodule classification using Taguchi-based convolutional neural networks for computer tomography images. *Electronics* **2020**, *29*, 1066. [CrossRef]
33. Verma, V.; Muttoo, S.K.; Singh, V.B. Multiclass malware classification via first- and second-order texture statistics. *Comput. Secur.* **2002**, *97*, 101895. [CrossRef]