information

Article

An Efficient Malware Classification Method Based on the AIFS-IDL and Multi-Feature Fusion

Xuan Wu [1] and Yafei Song [2,*]

[1] School of Postgraduate School, Air Force Engineering University, Xi'an 710051, China
[2] School of Air and Missile Defense, Air Force Engineering University, Xi'an 710051, China
* Correspondence: yafei_song@163.com

Abstract: In recent years, the presence of malware has been growing exponentially, resulting in enormous demand for efficient malware classification methods. However, the existing machine learning-based classifiers have high false positive rates and cannot effectively classify malware variants, packers, and obfuscation. To address this shortcoming, this paper proposes an efficient deep learning-based method named AIFS-IDL (Atanassov Intuitionistic Fuzzy Sets-Integrated Deep Learning), which uses static features to classify malware. The proposed method first extracts six types of features from the disassembly and byte files and then fuses them to solve the single-feature problem in traditional classification methods. Next, Atanassov's intuitionistic fuzzy set-based method is used to integrate the result of the three deep learning models, namely, GRU (Temporal Convolutional Network), TCN (Temporal Convolutional Network), and CNN (Convolutional Neural Networks), which improves the classification accuracy and generalizability of the classification model. The proposed method is verified by experiments and the results show that the proposed method can effectively improve the accuracy of malware classification compared to the existing methods. Experiments were carried out on the six types of features of malicious code and compared with traditional classification algorithms and ensemble learning algorithms. A variety of comparative experiments show that the classification accuracy rate of integrating multi-feature, multi-model aspects can reach 99.92%. The results show that, compared with other static classification methods, this method has better malware identification and classification ability.

Keywords: intuitionistic fuzzy set; deep learning; malware classification; multi-feature fusion

Citation: Wu, X.; Song, Y. An Efficient Malware Classification Method Based on the AIFS-IDL and Multi-Feature Fusion. *Information* **2022**, *13*, 571. https://doi.org/10.3390/info13120571

Academic Editor: Krzysztof Szczypiorski

Received: 20 September 2022
Accepted: 1 December 2022
Published: 9 December 2022

1. Introduction

With the continuous development of information technology, cyber security has become more important, and the number of cyberattacks has increased exponentially. Since the development of the first virus, named the Morris worm, in the 1980s, the international community has paid great attention to cybersecurity. However, aiming to the increasingly fast development of malware, virus makers have introduced polymorphism to avoid virus detection by constantly modifying and obfuscating the malware. As a result, the same type of malicious code can appear to be different to the malware-detection. The variability and invisibility of malware have posed great challenges to the prevention and control of cyberattacks. Therefore, understanding how to classify a large number of malicious codes quickly and accurately, and protect a cyber network, have remained a key challenge.

Malicious code classification divides malicious samples into different classes. Malware analysis methods can be divided into dynamic and static methods. Static analysis methods generally use reverse engineering technology to extract features to build models. The extracted features mostly include strings [1], opcodes [2], executables [3], and call graphs [4]. For instance, Santos et al. [5] performed unknown malware detection by calculating the frequency of opcodes. Kang et al. [6] extracted the opcode sequence from a disassembly file to represent the timing of malicious code execution and then used an N-gram algorithm to

characterize the opcode sequence. Nataraj et al. [7] proposed, for the first time, to convert malicious code executables into two-dimensional grayscale images and use the image texture features with a certain similarity in each malware class as features for malware detection model training. In recent years, image features have been widely used in malware detection. Gibert et al. [8] extracted two types of sequence features, namely byte sequence and opcode sequence, to classify malware using a convolutional neural network (CNN). The main advantage of static analysis methods is that they try to understand the logical structure of malware code without executing the code, thus avoiding damage to a cyber network as a result of the analysis. However, these methods cannot accurately identify obfuscation and packer codes. Therefore, dynamic analysis methods have been proposed to extract malware features. Kwon et al. [9] proposed a method for malware classification with API calling functions based on a recurrent neural network (RNN). The API calling functions of nine malware classes obtained from dynamic analysis were used as a training set, and a long short-term memory (LSTM) model was employed for classification; the average classification accuracy was 71%. Kolosnjaji et al. [10] used the open-source sandbox to extract the call sequence features of malware and then employed a CNN to perform deep learning on malicious codes, achieving accurate malware classification. This method is innovative but has low efficiency in feature extraction and model training. In contrast, dynamic analysis refers to methods that execute malware in a secure and controlled environment to analyze its behavior. To a certain extent, dynamic methods have high recognition accuracy for obfuscated or packed malicious codes. However, rapid technology development has enabled malware developers to launch targeted countermeasures. Moreover, dynamic analysis requires a secure and controlled environment, which can be easily detected by malware, whereas performing the analysis in a real environment can be expensive. Finally, malicious data are typically massive. Compared with dynamic analysis, the time and resource costs of static analysis methods are much lower. Therefore, this study uses static analysis to extract malware features.

In the past few years, various deep learning-based algorithms have been developed in the field of natural language processing and other related fields. These algorithms have strong learning capabilities and can mine data structures in high-dimensional data. Currently, deep learning-based malware detection has been a hot research topic. Many deep learning-based models, such as Convolution Neural Network (CNN), Recurrent Neural Network (RNN), and gated recurrent unit (GRU), can be used for malware classification. In addition, ensemble learning is a learning method that uses a series of learners and weights the results of the classifiers based on a certain rule, thus obtaining a model with better performance than a single classifier. The fuzzy set (FS) overcomes the constraints of binary logic in a classical set so that computers can also play a role in solving fuzzy problems with unclear extensions. Intuitionistic FS (IFS) has introduced the concept of the hesitation index, which has a more accurate representation of fuzzy phenomena than FS. Further, Atanassov's Intuitionistic FSs (AIFSs) [11] relax the condition that the sum of non-membership and membership equals one. An important application of AIFSs is multi-attribute group decision making (MAGDM) [12]. The classifier ensemble method is similar to the MAGDM, so the MAGDM method based on the IFS can be used to ensemble of classifier results. Through the ensemble of different deep learning models, the learning ability for the latent information of features can be improved. In ensemble learning methods, the input features are generally single features. Using different techniques, such as packing obfuscation, may have different effects on features.

To improve the feature mining ability, increase malware classification accuracy, decrease the interference of malware variants, packing, and obfuscation techniques, and reduce the cost of dynamic analysis, this study proposes an IFS-based method, which is named the Atanassov intuitionistic FSs-integrated deep learning, (AIFS-IDL).

The main contributions of this work can be summarized as follows:

1. A static malware classification method that integrates multiple features is proposed. The proposed method extracts six malware features from the disassembly and byte

files and integrates the advantages of different features to improve the classification accuracy;
2. Feature extraction capability of TCN for temporal data is introduced to fully learn the dependency relationship among data; The nonlinear fitting ability of GRU is used after information in the sequence, extracting the malware features based on the time series to improve the model classification effect; CNN has the characteristics of simple structure and low complexity. TCN, GRU, and CNN are used to learn the information on the extracted features fully;
3. The IFS and MAGDM methods are introduced to integrate the classification results and optimize the uncertainty, thus improving the classification accuracy and generalization ability of the deep learning algorithm.

The remainder of this paper is organized into four sections. Section 2 presents the research background and related work. Section 3 describes the proposed model in detail. Section 4 presents the experimental results and analysis. Section 5 gives conclusions and future prospects.

2. Related Work and Technology

2.1. IFS and MAGDM

The IFS method [11] was first proposed by Atanassov in 1986 and has been the most influential expansion of Zadeh's FS theory. The FS can describe the fuzzy concept of "one and the other". On the basis of the FS membership, IFS proposes a non-membership and hesitation index, which can further describe the neutral state of "neither one nor the other". Thus, IFS can accurately and comprehensively reflect the fuzzy phenomena of the real world.

Definition 1 [13]. *Assume that a non-empty set* $X = \{x_1, x_2, \cdots, x_n\}$ *is a given domain of discourse; then, an FS* A *in* X *is defined as follows:*

$$A = \{\langle x, \mu_A(x)\rangle | x \in X\} \tag{1}$$

where $\mu_A : X \to [0,1]$ *is the membership degree of* X *on* A.

Definition 2 [11]. *An IFS* B *in* $X = \{x_1, x_2, \cdots, x_n\}$ *is defined in Atanassov as*

$$B = \{\langle x, \mu_B(x), v_B(x)\rangle | x \in X\} \tag{2}$$

where $\mu_B : X \to [0,1]$ *and* $v_B : X \to [0,1]$ *are membership and non-membership degrees, respectively, and* $0 \leq \mu_B(x) + v_B(x) \leq 1$.

The *IFS* B defined in Atanassov can be expressed as π_B. Then, for $\forall x \in X$, the calculation of the hesitation index is as follows:

$$\pi_B(x) = 1 - \mu_B(x) - v_B(x) \tag{3}$$

Obviously, $\pi_B(x) \in [0,1]$, and $\forall x \in X$; $\pi_B(x)$ is known as an intuition index of x to B; $\pi_B(x)$ denotes a fuzzier. When $\pi_B(x) = 0$ and $\forall x \in X$, AIFS becomes an ordinary FS.

Since the introduction of AIFS, many studies have been devoted to exploring its applications and mathematical mechanisms. An important application of AIFSs is MAGDM [14–17]. However, in MAGDM problems, due to limited knowledge of experts and time pressure, information the AIFSs provide when evaluating a scheme may be uncertain or incomplete. Therefore, an appropriate model should be established to describe incomplete information. The AIFSs can be used to describe uncertainty caused by ambiguity and lack of knowledge by introducing the hesitation index. In addition, incomplete information can be directly pooled using intuitive fuzzy aggregation operators [18]. Therefore, AIFSs have been considered an effective tool to solve MAGDM problems and have attracted great research attention in solving

MAGDM problems due to a series of open topics in this field, such as the determination of attribute weights, efficient AIFSs, and incomplete information-based IFS.

Assume that $A = \{A_1, A_2, \cdots, A_n\}$ is a set of candidate schemes, $C = \{C_1, C_2, \cdots, C_m\}$ is a decision-making index, and $E = \{E_1, E_2, \cdots, E_s\}$ is a set of weights of decision makers. The weight of attribute A_i is w_i, where $i = 1, 2, \cdots, n$. The weights are represented by a weight vector, which is given by $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_n)^T$. Further, each decision maker is assigned a weighting factor λ_j, where $j = 1, 2, \cdots, s$. When solving MAGDM problems, the superiority of two schemes is typically used to represent the preference relationship [19]. This method is simpler than using numerical magnitudes to evaluate a single scheme [20]. The IFS theory constructs an intuitionistic fuzzy judgment matrix by comparing the preference between two schemes.

For $C = \{C_1, C_2, \cdots, C_m\}$, the intuitionistic fuzzy judgment matrix is given by:

$$R_k = \begin{array}{c} \\ C_1 \\ C_2 \\ \vdots \\ C_m \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \cdots & C_n \end{array} \\ \begin{pmatrix} \langle 0.5, 0.5 \rangle & \left\langle \mu_{12}^k, v_{12}^k \right\rangle & \cdots & \left\langle \mu_{1n}^k, v_{1n}^k \right\rangle \\ \left\langle \mu_{21}^k, v_{21}^k \right\rangle & \langle 0.5, 0.5 \rangle & \cdots & \left\langle \mu_{2n}^k, v_{2n}^k \right\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \left\langle \mu_{m1}^k, v_{m1}^k \right\rangle & \left\langle \mu_{m2}^k, v_{m2}^k \right\rangle & \cdots & \langle 0.5, 0.5 \rangle \end{pmatrix} \end{array} \tag{4}$$

where $r_{ij}^k = \left\langle \mu_{ij}^k, v_{ij}^k \right\rangle$ is an intuitionistic fuzzy value (IFV); μ_{ij}^k represents the preference of C_i over C_j; v_{ij}^k denotes the preference of C_j over C_i; $\mu_{ij} : X \to [0, 1]$ and $v_{ij} : X \to [0, 1]$.

When $\mu_{ij}^k = v_{ji}^k = 0.5$, C_i is the same as C_j, and $\mu_{ij}^k + v_{ji}^k \leq 1$. Based on the rules in the calculation matrix, R^* can be obtained, and the optimal and worst schemes in C denoted by C_{best} and C_{worst}, respectively, are determined.

In addition, consistency should be analyzed when constructing the intuitionistic fuzzy judgment matrix, since the lack of consistency may lead to misleading results in the MCGDM problem with intuitionistic fuzzy preference relation matrix [21].

A matrix R, $r_{ij}^k = \left\langle \mu_{ij}^k, v_{ij}^k \right\rangle$ is expressed as follows:

$$\begin{cases} \frac{1}{2}(1+\log_9 \mu_{best,i}) \times \frac{1}{2}(1+\log_9 \mu_{i,j}) = \frac{1}{2}(1+\log_9 \mu_{best,j}) \\ \frac{1}{2}(1+\log_9 \mu_{i,j}) \times \frac{1}{2}(1+\log_9 \mu_{j,worst}) = \frac{1}{2}(1+\log_9 \mu_{i,worst}) \end{cases} \tag{5}$$

$$\begin{cases} \frac{1}{2}(1+\log_9 \nu_{best,i}) \times \frac{1}{2}(1+\log_9 \nu_{i,j}) = \frac{1}{2}(1+\log_9 \nu_{best,j}) \\ \frac{1}{2}(1+\log_9 \nu_{i,j}) \times \frac{1}{2}(1+\log_9 \nu_{j,worst}) = \frac{1}{2}(1+\log_9 \nu_{i,worst}) \end{cases} \tag{6}$$

where $\mu_{best,worst} \in [0.5, 1]$ and $\nu_{best,worst} \in [0, 0.5]$.

If Equations (5) and (6) hold, the intuitionistic fuzzy judgment matrix satisfies strict consistency, but when these equations conflict, the consistency degree will be reduced. For μ_{ij}^k and v_{ij}^k, there are special cases where $\mu_{best,j} = \mu_{j,worst} = \mu_{best,worst}$ and $\nu_{best,j} = \nu_{j,worst} = \nu_{best,worst}$. In these cases, two new consistency indicators ω and δ are introduced, which can be obtained based on the value ranges of $\mu_{best,worst}$ and $\nu_{best,worst}$. Then, Equations (5) and (6) can be re-written as follows:

$$\begin{cases} (\frac{1}{2}(1+\log_9 \mu_{best,i}) - \omega) \times (\frac{1}{2}(1+\log_9 \mu_{i,j}) - \omega) = \frac{1}{2}(1+\log_9 \mu_{best,j}) + \omega \\ (\frac{1}{2}(1+\log_9 \mu_{i,j}) - \delta) \times (\frac{1}{2}(1+\log_9 \mu_{j,worst}) - \delta) = \frac{1}{2}(1+\log_9 \mu_{i,worst}) + \delta \end{cases} \tag{7}$$

The intuitionistic fuzzy preference judgment matrix is not necessarily consistent. In order to meet the consistency requirements as much as possible, we introduce two mathematical models. The mathematical models are model 4 and model 5 in literature [21].

A mathematical model $\min\xi$ is established to obtain a membership degree, which satisfies the following properties:

(DP1) $\left| \frac{\phi_{best}}{\phi_j} - 9^{(2 \times \mu_{best,j} - 1)} \right| \leq \xi$;

(DP2) $\left| \frac{\phi_j}{\phi_{worst}} - 9^{(2\times\mu_{worst}-1)} \right| \leq \xi$;
(DP3) $\sum_{j=1}^{n} \phi_j = 1$;
(DP4) $\phi_{best} \geq \cdots \geq \phi_j \geq \cdots \geq \phi_{worst}$;
(DP5) $\phi_j \geq 0, \xi \geq 0$

Similarly, in terms of the non-membership degree, a mathematical model minζ is established. The following model can be constructed:

(DP1) $\left| \frac{\varphi_{best}}{\varphi_j} - 9^{(2\times\nu_{best,j}-1)} \right| \leq \zeta$;
(DP2) $\left| \frac{\varphi_j}{\varphi_{worst}} - 9^{(2\times\nu_{worst}-1)} \right| \leq \zeta$;
(DP3) $\sum_{j=1}^{n} \varphi_j = 1$;
(DP4) $\varphi_{best} \geq \cdots \geq \varphi_j \geq \cdots \geq \varphi_{worst}$;
(DP5) $\varphi_j \geq 0, \zeta \geq 0$

Based on the two mathematical models, $(\phi_1, \phi_2, \cdots, \phi_{\mathrm{m}})^{\mathrm{T}}$ and $(\varphi_1, \varphi_2, \cdots, \varphi_{\mathrm{m}})^{\mathrm{T}}$ can be obtained, and the weight of the decision-making index $C = \{C_1, C_2, \cdots, C_m\}$ can be determined.

Based on the two mathematical models, the optimal solutions $(\phi_1, \phi_2, \cdots, \phi_{\mathrm{m}})^{\mathrm{T}}$ and $(\varphi_1, \varphi_2, \cdots, \varphi_{\mathrm{m}})^{\mathrm{T}}$ can be obtained, and the weight of the decision-making index can be determined. The reasonableness of weight is judged based on the consistency ratio $CR = \max\left\{\frac{\xi}{\omega}, \frac{\zeta}{\delta}\right\}$. The CR value ranges from zero to one. The smaller the value, the more reasonable the constructed intuitionistic fuzzy judgment matrix will be. When the constructed matrix is assessed as reasonable, the decision matrix is constructed to obtain the sequences of alternative schemes $W = \{\omega_1, \omega_2, \cdots, \omega_m\}^{\mathrm{T}} = ((\phi_1, \varphi_1), (\phi_2, \varphi_2), \cdots, (\phi_{\mathrm{m}}, \varphi_m))^{\mathrm{T}}$.

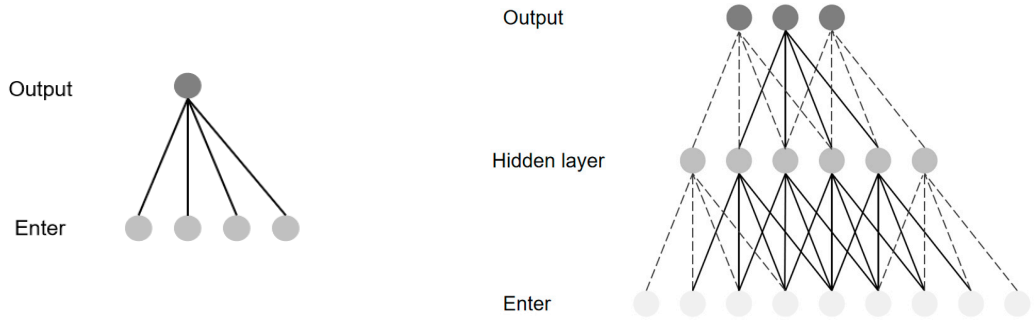
2.2. Deep Learning Models

A CNN is a deep feedforward neural network with sparse connectivity and weight-sharing characteristics. It consists of the input layer, convolutional layer, pooling layer, fully-connected layer, and output layer [22,23].

A temporal convolutional network (TCN) is a CNN-based network proposed by Bai et al. [24] for processing time-series data. In the TCN, causal convolution is added to the time-series convolutional network so that the upper and lower layers have a causal relationship, while dilated convolution and residual connection are used to address the gradient disappearance problem in RNNs. The TCN model can not only maintain a large receptive field for data, but can also reduce the amount of calculation and thus can efficiently control the memory length of a model and improve the accuracy of time-series data classification [25].

Compared with an ordinary one-dimensional CNN, TCN mainly has improvements in three aspects, which are as follows:

(a) Causal convolution: An output at time t is related only to the input at time t and the input of the previous layer [26]. Traditional CNN can see future information, whereas causal convolution can see only past information. Therefore, the causal convolution has a very strict time constraint and represents a one-way structure. A single causal convolution structure is shown in Figure 1a, and the overall structure is shown in Figure 1b, for a convolution kernel number off four. Using four convolution kernels means that four points are sampled from the previous layer as the input of the next layer;



(**a**) (**b**)

Figure 1. Causal convolution. (**a**) A single causal convolution structure, and (**b**) the overall structure.

(b) Dilated Convolution: With the increase in the number of dilated convolution layers, the expansion coefficient increases exponentially, and the increase in the receptive field of a layer will reduce the number of convolution layers. This reduces the amount of calculation and simplifies the network structure. In view of the traditional neural networks problem that time-series data modeling can only be extended by linearly stacking multi-layer convolutions, TCN uses dilated convolution to increase the receptive field of a layer to reduce the number of convolutional layers [27]. The network structure for a convolution kernel number of four and an expansion coefficient of one is shown in Figure 2. When the expansion coefficient of the input layer is one, the model samples data from the previous layer with an interval of one and inputs them to the next layer.

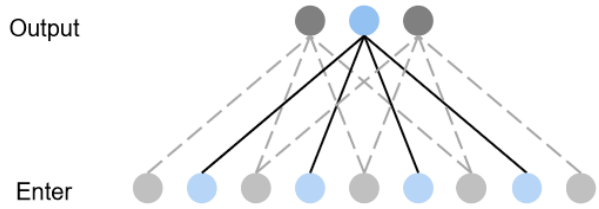


Figure 2. Dilated Convolution.

The difference between dilated and ordinary convolutions is that, in contrast to ordinary convolution, the dilated convolution permits interval sampling during convolution, and the sampling rate depends on the expansion coefficient.

The receptive field can be expressed as follows:

$$RF = (K - 1) * \mathrm{d} + 1 \tag{8}$$

where K is the convolution kernel size, and d is the expansion coefficient.

There are two methods that TCNs use to increase the receptive field. One method is to increase the expansion coefficient, and another method is to select a large convolution kernel value. In dilated convolution, the expansion coefficient increases exponentially with the network depth, so fewer layers are needed to obtain a large receptive field;

(c) Residual block: Residual block is an important part of the TCN structure. As shown in Figure 3, a residual block includes a dilated causal convolution layer and a nonlinear mapping layer and has an identity mapping method that connects layers, enabling the network to transmit information across layers. Residual connection can both increase the response and convergence speed of a deep network and solve the problem of slow learning speed caused by complex layer structure. Moreover, dropout and batch normalization are added to the residual block to prevent model overfitting and increase the training speed [28].

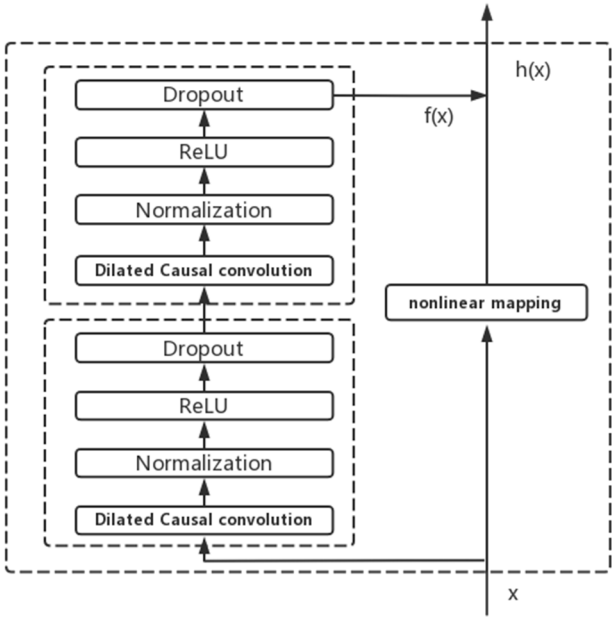


Figure 3. Residual block.

The residual connection converts input x to output $f(x)$ through a series of modules, which can be expressed by:

$$f(x) = h(x) - x(1) \tag{9}$$

A GRU, a variant of RNN, has a recursive structure and includes a memory function for processing time-series data. A GRU can effectively solve the problems of gradient disappearance and explosion that may occur in RNN training, thus effectively addressing the long-term memory problem. An LSTM network is also a variant of RNN [29]. It has similar performance to GRU, but GRU has a simpler structure, lower calculation complexity, and higher training efficiency [30]. The internal structure of GRU is shown in Figure 4, where the GRU has two inputs, which denote the output data of the previous moment and the input sequence of the current moment. The GRU output represents the state at the current moment. The GRU updates the model state using the reset and update gates, where the reset gate controls the forgetting degree of historical state information so that the network can eliminate unimportant information. The update gate controls the proportion of past information in the current state, helping to maintain long-term information [31].

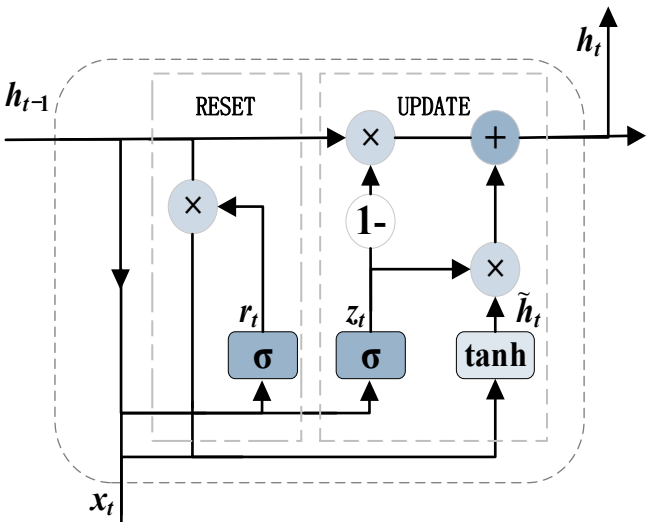


Figure 4. Gated Recurrent Unit.

The sigmoid activation function is defined by Equation (10) and presented in Figure 4. Its function is to convert the intermediate state into the range of [0, 1]. In Equation (12), h_{t-1} and h_t are the output states at times $(t-1)$ and t, respectively, x_t is the input sequence at time t, $\widetilde{\boldsymbol{h}}_{\boldsymbol{t}}$ is the candidate output state, W_r, W_z, $W_{\widetilde{h}}$, U_r, U_z, and $U_{\widetilde{h}}$ are the weight coefficient matrices corresponding to each part, tanh is the hyperbolic tangent function, and $\odot$ is the Hadamard product of a matrix.

$$\begin{cases} r_t = \sigma(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1}) \\ z_t = \sigma(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1}) \\ \widetilde{h}_t = \tan h(\boldsymbol{W}_{\widetilde{\boldsymbol{h}}} \boldsymbol{x}_t + \boldsymbol{U}_{\widetilde{\boldsymbol{h}}} (r_t \odot \boldsymbol{h}_{t-1})) \\ h_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \widetilde{h}_t \end{cases} \tag{10}$$

3. AIFS Malware Classification Based on Ensemble Deep Learning

The malware classification process includes three stages: preprocessing, feature extraction and training, and classification. The classification model structure is presented in Figure 5.

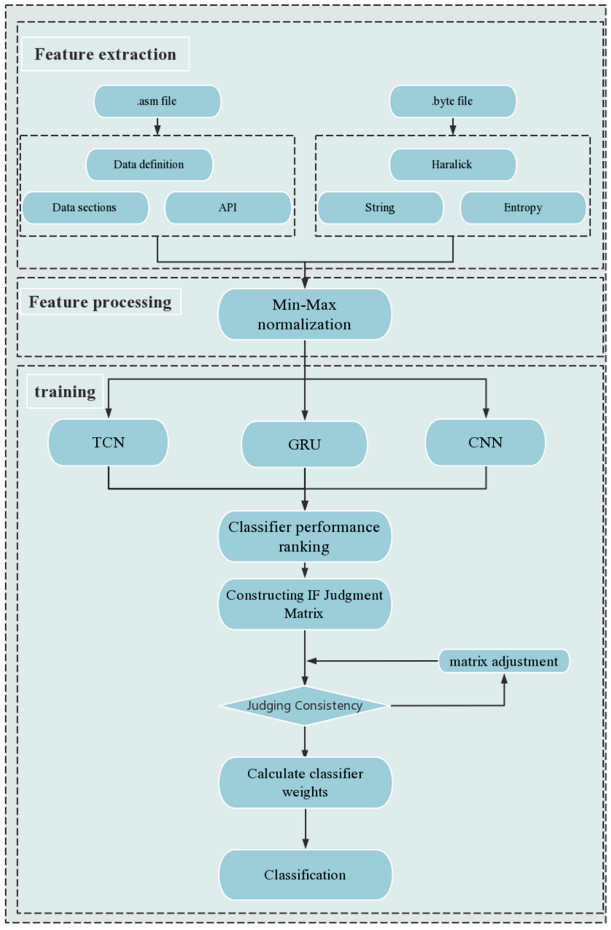


Figure 5. The malware classification model structure.

As shown in Figure 5, the proposed AIFS algorithm based on ensemble deep learning mainly includes the following steps. First, eight types of features are extracted from the two types of samples in the original dataset. Next, the original data features are subjected to the min–max normalization, and their labels are one-hot encoded. Then, the GRU, TCN, and CNN classifiers are used to train and predict the single and fused features, and the classification accuracy of each model is calculated. Afterward, the intuitionistic fuzzy judgment matrices of the three models are constructed based on the classification accuracy

and their consistency is evaluated. If a matrix is inconsistent, it will be corrected. Otherwise, the classifiers are ranked, and the weight of each classifier is calculated. Finally, based on the model weights and the MAGDM method, the classification results are obtained.

3.1. Feature Extraction

During malware classification, it is necessary to extract multiple features from data samples first and then input the extracted features into the classification model. Thus, in this study, six static features are extracted from the disassembly and byte files for malware classification, including image texture, string, and entropy features are extracted from the byte files. Data section, data definition, and API features are extracted from the disassembly files. The features are described in Figure 6.

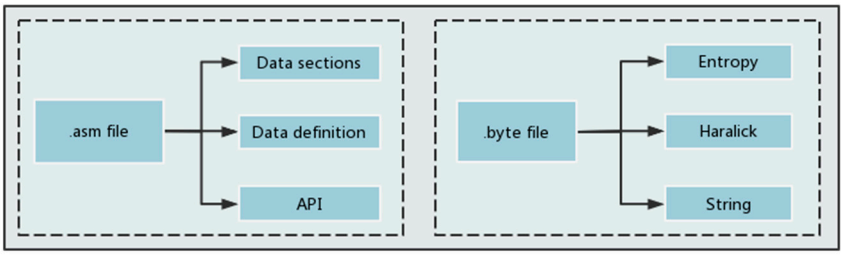


Figure 6. Feature description.

1. Data sections

The sections of the .asm file are mainly used to store information such as code, data, and resources, whereas .asm files are composed of some predefined sections such as .text (code section) and .data (data section), as shown in Table 1.

Table 1. Structure of .asm files.

Name	Description
.text	Code section; program code segment identifier
.data	Data section; initialize data segment, store global data, and global constants
.bss	Uninitialized data segment; store global data and global constants
.rdata	Resource data segment
.edata	Export table; addresses of exported external functions
.idata	Import table; addresses of imported external functions
.rsrc	Resource section; store program resources such as icons and menus
.tls	Store pre-stored thread-local variables, including initialization data, callback functions for each thread initialization and termination, and TLS index
.reloc	Base address relocation table; all content in the mirror that needs to be relocated

Data in a data section can be modified, reordered, and new structures can be created using check-avoiding techniques such as packing. In this study, the number of lines of each section is calculated, as well as the total number of lines of all sections, the number of lines of unknown sections, and the proportion of each section. The total number of lines of each section is referred to as total_[section]. For instance, total_.bss denotes the total number of lines in the .bss section. The proportion of each section to all sections is denoted by [section]_por, and .bss_por represents the percentage of .bss section. The overall results are shown in Table 2.

Table 2. List of features in the section category.

Name	Description
Num_Sections	Total number of sections
Unknown_Sections	Number of unknown sections
Unknown_Sections_lines	Number of lines in unknown sections
known_Sections_por	Proportion of known sections
Unknown_Sections_por	Proportion of unknown sections
Unknown_Sections_lines_por	Proportion of lines in unknown sections
Unknown_Sections_lines_por	Proportion of lines in unknown sections

2. Data definition

This functional class differs among malware samples, and some samples may not contain any API calls due to packaging or contain only a small number of operational codes. Particularly, due to packaging, some samples mostly contain db, dw, and dd instructions that are used to set bytes, words, and double words, respectively. In addition, some malicious codes are packaged or packed, and the .asm files do not contain normal code segments. For instance, in the malicious code 58kxhXouHzFd4g3rmInB, there are only two data segments, HEADER and GAP. In the sample 3r0swJ67FWm5HXDnjaIy, there is only one segment, the seg000 segment. The sample 6tfw0xSL2FNHOCJBdlaA contains only the .gap segment. In the malicious sample 36eMEj40inf8r5vVQlBx, after UPX compression, the disassembled .text segments become UPX* code segments, and most disassembled codes contain only data definition instructions, such as db.

Accordingly, through the analysis, it is found that there are three main instructions in the packed samples considered in this study, namely, db, dd, and dw instructions. In addition, by calculating the proportion of the three instructions in the sample and different data sections under the frequency of zero, features of packed samples are identified, as well as features of different malware classes after packing. Based on the analysis of these features, it is expected to improve the classification result of packed samples.

3. API features

The APIs are mainly stored in the .idata segment. The total number of APIs is very large, and APIs bring little or no meaningful information to malware classification. Based on the analysis of nearly 500 K malware samples, the analysis is limited to the top 794 most common APIs used in malicious binaries. The descriptions of some of the APIs are shown in Table 3. For the API features, the feature occurrence frequency is calculated for each sample, and the influence of APIs in different malware classes on the classification result is I.

Table 3. API interface example.

Name	Description
API_GetProcAddress	Retrieves the output library function address from the specified dynamic link library (DLL).
API_LoadLibraryA	Loads the specified module into the address space of the calling process. The specified module may cause other modules to be loaded.
API_GetModuleHandleA	Retrieves a module handle for the specified module. The module must have been loaded by the calling process.
API_ExitProcess	Ends the calling process and all its threads.
API_VirtualAlloc	Reserves, commits, or changes the state of a region of pages in the virtual address space of the calling process. Memory allocated by this function is automatically initialized to zero.
API_WriteFile	Writes data to the specified file or input/output (I/O) device.

4. Entropy feature

Entropy is a metric used to measure disordered quantities and can also be used to detect possible obfuscation. In this study, entropy has a value between zero and eight, and is computed from the byte-level representation of each malware sample. The goal is to measure the disorder degree in the distribution of bytes. First, a sliding window method is used to express malware as a series of entropy measures, which is denoted by $E = e_i : i = 1, \ldots, N$ where e_i is the entropy value measured in a window i and N is the number of windows. Then, the entropy is calculated using the Shannon entropy formula as follows:

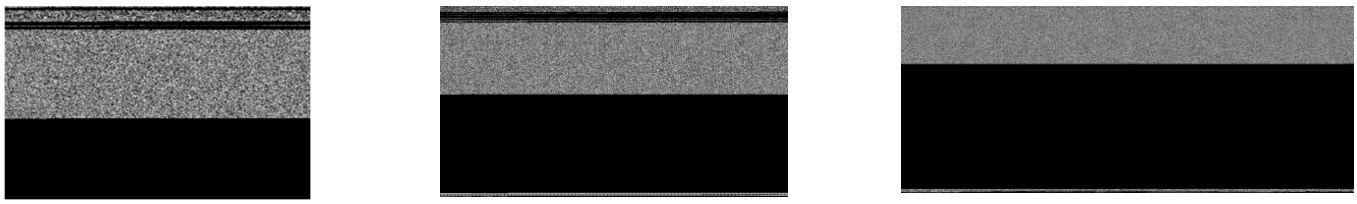
$$e_i = -\sum_{j=1}^{m} p(j) \log_2 p(j) \tag{11}$$

where $p(j)$ is the frequency of byte j in a window i, and m is the number of different bytes in the window.

Then, the entropy sequence obtained using the sliding window method is calculated, which is the entropy for a window of 10,000 bytes. Different statistical measures such as quantiles, percentiles, mean, and variance are used to analyze the results. Finally, the entropy of all bytes in malware is calculated.

5. Haralick features

The Haralick features are statistical features that are computed over the entire image. They are texture features based on an adjacency matrix. Namely, the adjacency matrix stores a number of pixels with a value of i in a position (i, j), which are adjacent to pixels with a value of j. Using different definitions, it is possible to obtain features with slight variations. Generally, the average values in all directions are calculated to obtain rotational invariance. Moreover, representing malware as an image can cause problems, as shown in Figure 7, where the textures of the two images are almost the same, even though the two samples belong to different classes. Hence, other features are needed to improve the classification accuracy.



(**a**) (**b**) (**c**)

Figure 7. Malware sample image. (**a**) Vundo family. (**b**) Simda family. (**c**) Tracur family.

6. String features

Visible strings length: The hex dump of each PE is used to extract possible ASCII strings. The length, frequency, distribution, and other information of various types of visible strings are counted. It is worth noting that some meaningful visible strings deserve more in-depth study [32]. Since the feature can extract many valueless strings, it is not reasonable to use string length directly. To reduce noise and avoid overfitting, this study uses only the distribution histogram of string lengths.

3.2. IFS-MAGDM

Two crucial tasks in the proposed method are the intuitionistic fuzzy judgment matrix construction and the classifier weight calculation.

By comparing the classification accuracy of different classifiers {*GRU*, *TCN*, *CNN*} = {*C1*, *C2*, *C3*}, the intuitionistic fuzzy judgment matrix R is constructed using $\mu_{ij} = \frac{\text{Accuracy of } C_i}{\text{Accuracy of } C_i+C_j}$ and $\nu_{ij} = \frac{\text{Accuracy of } C_j}{\text{Accuracy of } C_i+C_j}$ as follows:

$$R = \begin{array}{c} \\ GRU \\ TCN \\ CNN \end{array} \begin{array}{c} \begin{array}{ccc} GRU & TCN & CNN \end{array} \\ \begin{pmatrix} \langle 0.5, 0.5 \rangle & \left\langle \mu_{12}^k, v_{12}^k \right\rangle & \left\langle \mu_{1n}^k, v_{1n}^k \right\rangle \\ \left\langle \mu_{21}^k, v_{21}^k \right\rangle & \langle 0.5, 0.5 \rangle & \left\langle \mu_{2n}^k, v_{2n}^k \right\rangle \\ \langle \mu_{31}, v_{31} \rangle & \left\langle \mu_{m2}^k, v_{m2}^k \right\rangle & \langle 0.5, 0.5 \rangle \end{pmatrix} \end{array} \tag{12}$$

where $R_{12} = (\mu_{12}, \nu_{12})$ represents the difference in accuracy of the *GRU* model over the *TCN* model, and $R_{ij}^* = \left(\mu_{ij}^*, \nu_{ij}^*\right)$ is obtained based on the following rules:

1. When $\mu_{ij} > \nu_{ij}$, $\begin{array}{l} \mu_{i,j}^* = \mu_{i,j} + \nu_{i,j} \times 10\% \\ \nu_{i,j}^* = \nu_{i,j} - \mu_{i,j} \times 10\% \end{array}$;
2. When $\mu_{ij} = \nu_{ij}$, $\begin{array}{l} \mu_{i,j}^* = \mu_{i,j} \\ \nu_{i,j}^* = \nu_{i,j} \end{array}$;
3. When $\mu_{ij} < \nu_{ij}$, $\begin{array}{l} \mu_{i,j}^* = \mu_{i,j} - \nu_{i,j} \times 10\% \\ \nu_{i,j}^* = \nu_{i,j} + \mu_{i,j} \times 10\% \end{array}$.

Further, based on the matrix $R_{ij}^* = \left(\mu_{ij}^*, \nu_{ij}^*\right)$, the best and worst models in the classification model C, denoted by C_{best} and C_{worst}, respectively, are obtained. After passing the consistency test, the weights of classifiers are calculated by mathematical models.

4. Experimental Results

4.1. Experimental Setup

The performance of the proposed malware classification method based on the IFS for different features was verified by three experiments, which were as follows:

- Experiment 1: Single-feature comparison;
- Experiment 2: Multi-feature fusion comparison;
- Experiment 3: Comparison of different classification algorithms.

4.2. Evaluation Indices

This study adopted four common evaluation indices in the field of malware classification and detection, namely, accuracy (*Acc*), precision (*PR*), recall rate (*RR*), and F1-score (*F1*), which were respectively calculated by:

$$\begin{cases} Acc = \frac{TP+TN}{TP+TN+FP+FN} \\ PR = \frac{TP}{TP+FN} \\ RR = \frac{TP}{TP+FP} \\ F1 = \frac{2 \times PR \times RR}{PR+RR} \end{cases} \tag{13}$$

TP represents the number of true positives; *FN* is the number of false negatives; *FP* is the number of false positives; *TN* is the number of true negatives.

4.3. Hardware and Dataset

The computer used in the experiments operated on the Win10 system and included an Intel(R) Core™ i9-9880H CPU @ 2.30 GHz, 64-GB memory, and Quadro RTX 4000 GPU. The program was written in PyCharm2021.2.2 using Python3.7 and ran in the CUDA11.0

accelerated environment. The neural network model used the deep learning framework versions of TensorFlow2.4.1 and Keras2.4.3.

The experimental data were open-source and downloaded from Microsoft [33]. The malware in the dataset was divided into nine classes, and contained 10,868 samples. The dataset is shown in Figure 8, which shows that each sample file was available in two formats, .asm file format and .bytes file format. In this study, the assembly language .asm file was used.

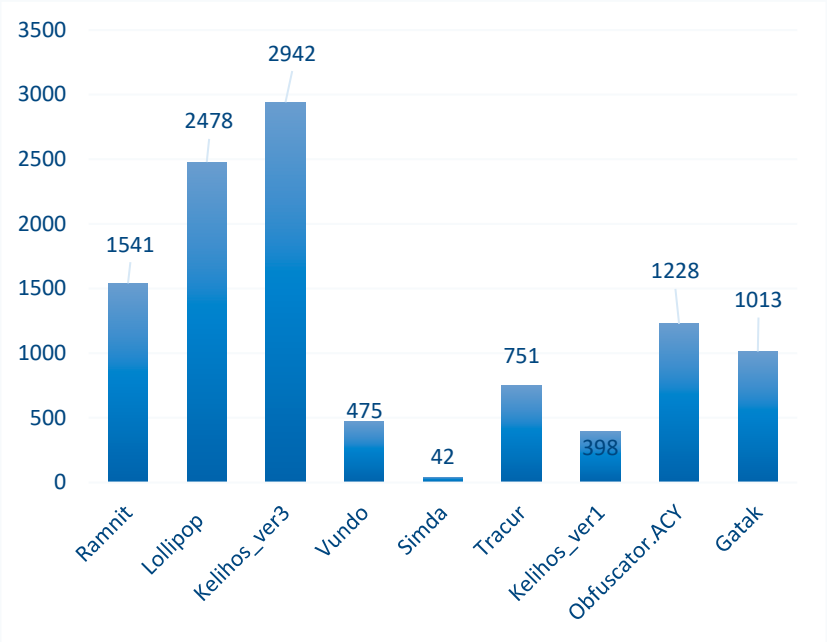


Figure 8. Microsoft malware data set and partition.

In the experiment, the dataset was divided into training and test sets according to the ratio of 7:3. The training set contained 7608 samples and the test set included 3260 samples. The parameter random_state was fixed to ensure that the divided dataset remained the same.

4.4. Single-Feature Comparison

Feature selection methods have been compared by authors in [34–36], which discusses how the methods affect the classification stage. This section observes the impact of different features on the classification results through experiments on individual features. Six classes of features were extracted, and the GRU, TCN, CNN, and AIFS-IDL models were used for classification. By using five-fold cross-validation, the average value of indices of each feature in the three experiments was calculated. For the six extracted features, the performances of three single-feature models were compared, as shown in Table 4 and Figure 9.

As presented in Table 4 and Figure 9, the proposed model achieved better classification performance than the other models. Compared with the TCN, GRU, and CNN models, the accuracy of the proposed AIFS-IDL model has improved on different single features extracted from this dataset. Based on the accuracy, precision, recall, and F1-score results of the models based on the six classes of features extracted on this dataset, the proposed AIFS-IDL model performed better than the TCN, GRU, and CNN. Thus, the results verified that the proposed AIFS-IDL model could improve malware classification performances.

Table 4. Single-feature comparison experiment results.

Features		Model	Accuracy	Precision	Recall	F1-Score
Features from the .asm file	Data section	GRU	97.79%	97.34%	97.41%	97.35%
		TCN	98.36%	97.81%	97.72%	97.72%
		CNN	98.42%	98.24%	98.16%	98.18%
		AIFS-IDL	98.81%	98.62%	98.52%	98.50%
	Data definition	GRU	98.47%	97.49%	97.50%	97.48%
		TCN	97.94%	97.62%	97.57%	97.58%
		CNN	98.23%	97.91%	97.90%	97.91%
		AIFS-IDL	98.50%	98.53%	98.52%	98.64%
	API	GRU	94.54%	94.18%	94.02%	94.02%
		TCN	77.31%	80.49%	76.45%	74.54%
		CNN	97.98%	97.74%	97.55%	97.57%
		AIFS-IDL	98.30%	98.25%	98.17%	98.22%
Features from the byte file	Entropy	GRU	97.77%	97.09%	97.20%	97.13%
		TCN	93.41%	92.38%	92.38%	92.18%
		CNN	98.91%	98.50%	98.60%	98.54%
		AIFS-IDL	99.01%	98.56%	98.67%	98.56%
	Haralick	GRU	95.29%	94.21%	94.33%	94.23%
		TCN	95.25%	94.23%	94.33%	94.24%
		CNN	94.89%	94.34%	94.42%	94.31%
		AIFS-IDL	95.32%	94.65%	94.76%	94.28%
	String	GRU	96.98%	96.97%	96.98%	96.94%
		TCN	92.75%	92.84%	92.75%	92.59%
		CNN	98.22%	97.81%	97.89%	97.85%
		AIFS-IDL	98.35%	98.03%	98.06%	97.97%

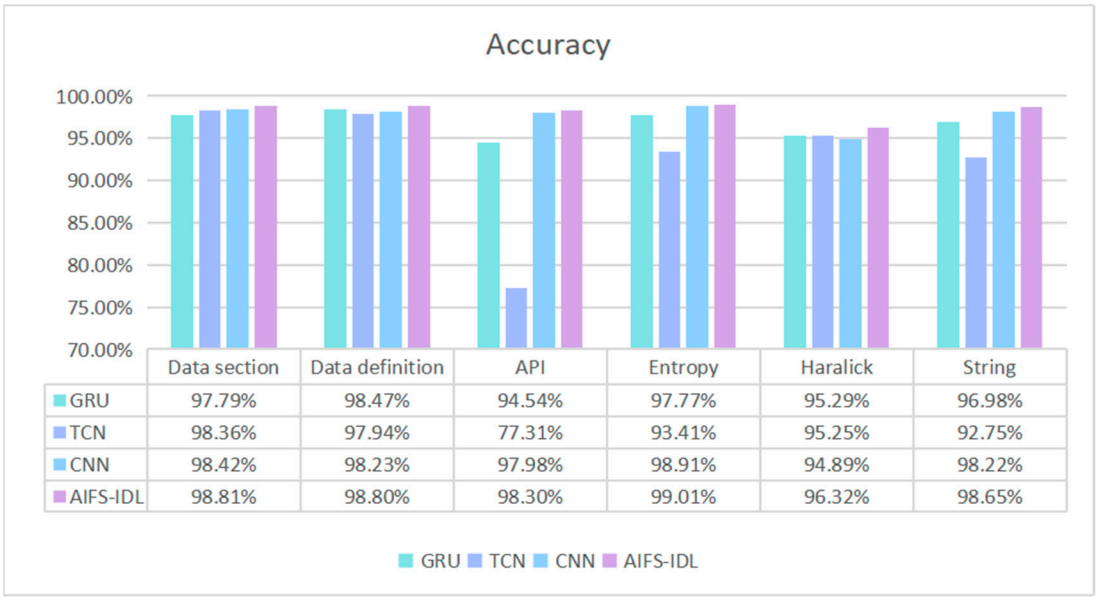


Figure 9. Accuracy of single feature.

The ensemble learning model had slightly higher or comparable values of the evaluation indices compared to the single-feature models, which demonstrated the strong classification capability of the proposed AIFS-IDL model.

4.5. Multi-Feature Fusion Comparison

Four different features in the .asm files and .byte files were extracted and then all features were fused. The three single-feature models were compared, and the experimental results are shown in Table 5 and Figure 10.

Table 5. Multi-feature fusion experiment results.

Features	Model	Accuracy	Precision	Recall	F1-Score
Features from the .asm file	GRU	99.63%	99.64%	99.63%	99.62%
	TCN	99.36%	99.36%	99.36%	99.36%
	CNN	99.36%	99.00%	98.99%	98.97%
	AIFS-IDL	99.84%	99.86%	99.85%	99.85%
Features from the .byte file	GRU	99.26%	99.27%	99.26%	99.25%
	TCN	98.07%	97.91%	98.07%	97.98%
	CNN	98.99%	98.91%	98.90%	98.88%
	AIFS-IDL	99.46%	99.47%	99.46%	99.45%
All features	GRU	99.72%	99.73%	99.72%	99.73%
	TCN	99.45%	99.46%	99.45%	99.46%
	CNN	99.45%	99.37%	99.36%	99.34%
	AIFS-IDL	99.92%	99.92%	99.92%	99.92%

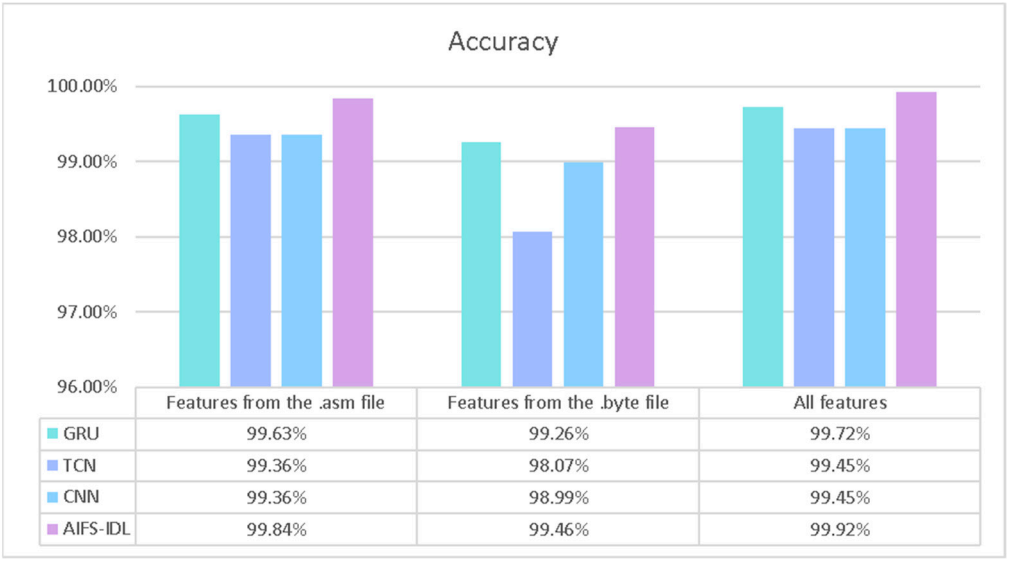


Figure 10. Accuracy of multi feature.

As presented in Table 5 and Figure 10, the proposed model achieved better classification performance than the other models. Compared with the TCN, GRU, and CNN models, the accuracy of the proposed AIFS-IDL model on the dataset was improved by 0.21%, 0.48%, and 0.48%, respectively. Based on the data set, all features are fused, and three features are fused based on .asm files or .byte files, Based on the accuracy, precision, recall, and F1 score results of three different feature models, the proposed AIFS-IDL model performed better than the TCN, GRU, and CNN. Thus, the results verified that the proposed AIFS-IDL model could improve malware classification performances.

The ensemble learning model had slightly higher or comparable values of the evaluation indices compared to the multi-feature models, which demonstrated the strong classification capability of the proposed AIFS-IDL model.

4.6. Comparison of Different Classification Algorithms

In order to evaluate the performance of the classification model, the model was compared with the model method proposed in [8,37–46]. The data sets used in these documents are the same as those in this paper.

Among the five models, two models were based on machine learning, one model was related to gene sequence classification, and the other two models were deep learning-based models. Gilbert et al. [8] first extracted byte and opcode sequences and then used two CNN classifiers for classification. Yan et al. [37] used a CNN model to extract features from grayscale images and employed an LSTM model to extract opcode features. Then, the features were fused and used for classification. Narayanan et al. [38] extracted the opcode and grayscale features and then used the support vector machine (SVM) for classification. Narayanan et al. [39] processed the grayscale images converted from malware, performed dimensionality reduction of the image features using the PCA, and finally used the K-nearest neighbor method for classification. Guo et al. [40] used the Strand method and other gene detection methods to classify texts. Ni. et al. [41] proposed the MCSC model. The opcode features in malware samples are extracted, converted to grayscale maps by Sim-Hash coding, and feature extraction and classification are performed using convolutional neural networks. Le. et al. [42] proposed a method to convert the original samples into binary features and train the binary sequence features using a deep learning model. Khan et al. [43] introduced a feature processing method that converts an opcode into a linear matrix, which is used to generate an image. The extracted image features were trained using ResNet and GoogleNet models. N. Marastoni et al. [44] uses CNN and LSTM models, respectively, to classify malware image features using migration learning methods. Darem et al. [45] proposed an approach with integrated deep learning, feature engineering, image transformation and processing techniques. Darem et al. [45] uses CNN and XGBoost methods. The experimental results are shown in Table 6.

Table 6. Comparison with the existing models.

Authors	Time	Method	Model	Features	Accuracy
Burnaev et al. [38]	2016	One-class SVM	SVM	Opcode + grayscale image	92%
Narayanan et al. [39]	2016	PCA and kNN	KNN	grayscale image	96.6%
Drew et al. [40]	2017	Strand Gene Sequence	Strand	asm sequence	98.59%
Ni et al. [41]	2018	Sim–Hash and NN	CNN	Grayscale images	98.86%
Le et al. [42]	2018	-	CNN, LSTM and RNN	Binary representation	98.20%
Yan et al. [37]	2018	MalNet	CNN and LSTM	Raw file data	99.36%
Khan et al. [43]	2019	-	ResNet and GoogleNet	Image	88.36%
Gibert et al. [8]	2020	Orthrus	CNN	Byte + Opcode	99.24%
Marastoni et al. [44]	2021	-	CNN and LSTM	Image-based data	98.5%
Darem et al. [45]	2021	ensemble	CNN and XGBoost	Opcode + image+ segment + other	99.12%
X et al. [46]	2022	TCN-BiGRU	TCN and BiGRU	Opcode + Byte sequence	99.72%
AIFS-IDL	current	AIFS-IDL	TCN, CNN, and GRU	Disassembly file + Byte file	99.92%

Although the methods mentioned above achieve good results, they are inferior compared to the method proposed in this study. To improve the classification accuracy, the proposed method fused six features extracted from the .asm files and .byte files and integrated the training results of the GRU, TCN, and CNN models based on AIFS and MAGDM. The accuracy of the proposed model reached 99.92%, outperforming that of the other models. Therefore, the proposed multi-feature AIFS-IDL method achieved the best performance among all methods.

5. Conclusions and Future Work

In this study, an AIFS-IDL and multi-feature fusion-based method is proposed for malware classification. The proposed method extracts three types of features from the

disassembly files and three types of features from the byte files in malware samples, accounting for a total of six types of features. The six features are extracted with different focuses. They compensate each other for the deficiencies of their own features. Then, according to the classification accuracy of the GRU, TCN, and CNN models and based on IFS theory, an intuitionistic fuzzy judgment matrix is constructed and used to determine the classifier weights. This method combines the advantages of different classifiers. Finally, the MAGDM method is used to classify malware samples. The experiments show that the proposed method can improve the classification accuracy and generalization ability of traditional deep learning models. In addition, the performance of the proposed model is compared with single-feature models, including the GRU, TCN, and CNN models, on malware datasets. This combines the advantages of different classifiers to avoid the losses generated by a single classifier in the process of feature extraction. The experimental results show that, compared to the traditional algorithms, the classification accuracy of the proposed method is better, and the generalization ability is improved to a certain extent.

The method proposed in this paper can meet the requirements of higher data volume level well, and has a series of advantages, such as high classification accuracy, anti-confusion, and shelling. However, the proposed method in this paper is requires higher computational time. Therefore, the next consideration would be to refer to [47–49] to study the time consumption and improve the model in a lightweight way. At the same time, we will continue to study to the anti-attack technology in the field of malware classification. We will also plan to use some newer anti-attack technology attack detection models to verify whether the classification method can resist more advanced anti-attack methods.

Author Contributions: Conceptualization, X.W. and Y.S.; methodology, X.W.; validation, X.W. and Y.S.; formal analysis, X.W.; writing-original draft preparation, X.W.; writing-review and editing, Y.S.; funding acquisition, Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Natural Science Foundation of China under grants No. 61703426, No. 61876189 and No. 61806219, and by the Post-Doctorial Science Foundation of China under Grant 2018M633680, and by by Young Talent fund of University Association for Science and Technology in Shaanxi, China, under Grant No. 20190108, and the Innovation Capability Support Plan of Shaanxi, China, under Grant No. 2020KJXX-065.

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

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

References

1. Zhao, J.; Zhang, S.; Liu, B.; Cui, B. Malware detection using machine learning based on the combination of dynamic and static features. In Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, China, 30 July–2 August 2018.
2. Yang, P.; Pan, Y.; Jia, P.; Liu, L. Research on malware detection based on vector features of assembly instructions. *Inf. Secur. Res.* **2020**, *6*, 113–121.
3. Raff, E.; Sylvester, J.; Nicholas, C. Learning the pe header, malware detection with minimal domain knowledge. In Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, Dallas, TX, USA, 3 November 2017; pp. 121–132.
4. Zhao, S.; Ma, X.; Zou, W.; Bai, B. DeepCG: Classifying metamorphic malware through deep learning of call graphs. In Proceedings of the International Conference on Security and Privacy in Communication Systems, Orlando, FL, USA, 23–25 October 2019; pp. 171–190.
5. Santos, I.; Brezo, F.; Ugarte-Pedrero, X.; Bringas, P.G. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Inf. Sci.* **2013**, *231*, 64–82. [CrossRef]
6. Kang, B.; Yerima, S.Y.; McLaughlin, K.; Sezer, S. N-opcode Analysis for Android Malware Classification and Categorization. In Proceedings of the 2016 International Conference on Cyber Security and Protection of Digital Services (Cyber Security), London, UK, 13–14 June 2016; pp. 1–7.
7. Nataraj, L.; Karthikeyan, S.; Jacob, G.; Manjunath, B.S. Malware images: Visualization and automatic classification. In Proceedings of the VizSec '11: Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburgh, PA, USA, 20 July 2011; Volume 4. [CrossRef]

8. Gibert, D.; Mateu, C.; Planes, J. Orthrus: A Bimodal Learning Architecture for Malware Classification. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
9. Kwon, I.; Im, E.G. Extracting the Representative API Call Patterns of Malware Families Using Recurrent Neural Network. In Proceedings of the International Conference, Krakow Poland, 20–23 September 2017; pp. 202–207.
10. Kolosnjaji, B.; Zarras, A.; Webster, G.; Eckert, C. Deep learning for classification of malware system call sequences. In Proceedings of the Australasian Joint Conference on Artificial Intelligence, Hobart, Australia, 5–8 December 2016; pp. 137–149.
11. Atanassov, K. Intuitionistic fuzzy sets. *Fuzzy Sets Syst.* **1986**, *20*, 87–96. [CrossRef]
12. Das, S.; Dutta, B.; Guha, D. Weight computation of criteria in a decision-making problem by knowledge measure with intuitionistic fuzzy set and interval-valued intuitionistic fuzzy set. *Soft Comput.* **2015**, *20*, 3421–3442. [CrossRef]
13. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [CrossRef]
14. Pal, N.R.; Bustince, H.; Pagola, M.; Mukherjee, U.K.; Goswami, D.P.; Beliakov, G. Uncertainties with Atanassov's intuitionistic fuzzy sets: Fuzziness and lack of knowledge. *Inf. Sci.* **2013**, *228*, 61–74. [CrossRef]
15. Nguyen, H. A new knowledge-based measure for intuitionistic fuzzy sets and its application in multiple attribute group decision making. *Expert. Syst. Appl.* **2015**, *42*, 8766–8774. [CrossRef]
16. Chen, S.M.; Tan, J.M. Handling multicriteria fuzzy decision making problems based on vague set theory. *Fuzzy Sets and Systems* **1994**, *67*, 163–172. [CrossRef]
17. Wu, X.; Song, Y.; Wang, Y. Distance-Based Knowledge Measure for Intuitionistic Fuzzy Sets with Its Application in Decision Making. *Entropy* **2021**, *23*, 1119. [CrossRef]
18. Wu, X.; Huang, F.; Hu, Z.; Huang, H. Faster Adaptive Federated Learning. *arXiv* **2022**, arXiv:2212.00974v1.
19. Xu, Z. A survey of preference relations. *Int. J. Gen. Syst.* **2007**, *36*, 179–203. [CrossRef]
20. Bustince, H.; Pagola, M.; Mesiar, R.; Hullermeier, E.; Herrera, F. Grouping, Overlap, and Generalized Bientropic Functions for Fuzzy Modeling of Pairwise Comparisons. *IEEE Trans. Fuzzy Syst.* **2011**, *20*, 405–415. [CrossRef]
21. Mou, Q.; Xu, Z.; Liao, H. A graph based group decision making approach with intuitionistic fuzzy preference relations. *Comput. Ind. Eng.* **2017**, *110*, 138–150. [CrossRef]
22. Xiang, Q.; Wang, X.; Song, Y.; Lei, L.; Li, R.; Lai, J. One-dimensional convolutional neural networks for high-resolution range profile recognition via adaptively feature recalibrating and automatically channel pruning. *Int. J. Intell. Syst.* **2021**, *36*, 332–361. [CrossRef]
23. Xiang, Q.; Wang, X.; Lai, J.; Song, Y.; Li, R.; Lei, L. Multi-scale group-fusion convolutional neural network for high-resolution range profile target recognition. *IET Radar Sonar Navig.* **2022**, *16*, 1997–2016. [CrossRef]
24. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**, arXiv:1803.01271.
25. Fan, Y.Y.; Li, C.J.; Yi, Q.; Li, B.Q. Classification of Field Moving Targets Based on Improved TCN Network. *Comput. Eng.* **2012**, *47*, 106–112.
26. Yating, G.; Wu, W.; Qiongbin, L.; Fenghuang, C.; Qinqin, C. Fault Diagnosis for Power Converters Based on Optimized Temporal Convolutional Network. *IEEE Trans. Instrum. Meas.* **2020**, *70*, 1–10. [CrossRef]
27. Huang, Q.; Hain, T. Improving Audio Anomalies Recognition Using Temporal Convolutional Attention Network. In Proceedings of the ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021. [CrossRef]
28. Zhu, R.; Liao, W.; Wang, Y. Short-term prediction for wind power based on temporal convolutional network. *Energy Rep.* **2020**, *6*, 424–429. [CrossRef]
29. Xu, Z.; Zeng, W.; Chu, X.; Cao, P. Multi-Aircraft Trajectory Collaborative Prediction Based on Social Long Short-Term Memory Network. *Aerospace* **2021**, *8*, 115. [CrossRef]
30. Liu, Y.; Ma, J.; Tao, Y.; Shi, L.; Wei, L.; Li, L. Hybrid Neural Network Text Classification Combining TCN and GRU. In Proceedings of the 2020 IEEE 23rd International Conference on Computational Science and Engineering (CSE), Guangzhou, China, 29 December 2020–1 Januray 2021; pp. 30–35.
31. Sun, Y.C.; Tian, R.L.; Wang, X.F. Emitter signal recognition based on improved CLDNN. *Syst. Eng. Electron.* **2021**, *43*, 42–47.
32. Digital Bread Crumbs: Seven Clues to Identifying Who's Behind Advanced Cyber Attacks [EB/OL]. Available online: https://www.techrepublic.com/resource-library/whitepapers/digital-bread-crumbs-seven-clues-to-identifying-who-is-behind-advanced-cyber-attacks/ (accessed on 1 August 2022).
33. Microsoft Malware Classification Challenge (Big 2015). 2015. Available online: https://www.kaggle.com/c/malware-classification/data (accessed on 13 April 2015).
34. Lee, J.; Pak, J.; Lee, M. Network intrusion detection system using feature extraction based on deep sparse autoencoder. In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea, 21–23 October 2020.
35. Injadat, M.N.; Moubayed, A.; Nassif, A.B.; Shami, A. Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection. *IEEE Trans. Netw. Serv. Manag.* **2020**, *18*, 1803–1816. [CrossRef]
36. Di Mauro, M.; Galatro, G.; Fortino, G.; Liotta, A. Supervised feature selection techniques in network intrusion detection: A critical review. *Eng. Appl. Artif. Intell.* **2021**, *101*, 104216. [CrossRef]

37. Yan, J.; Qi, Y.; Rao, Q. Detecting Malware with an Ensemble Method Based on Deep Neural Network. *Secur. Commun. Netw.* **2018**, *2018*, 7247095. [CrossRef]
38. Burnaev, E.; Smolyakov, D. One-Class SVM with Privileged Information and Its Application to Malware Detection. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; pp. 273–280. [CrossRef]
39. 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, 25–29 July 2017; pp. 338–342. [CrossRef]
40. Drew, J.; Hahsler, M.; Moore, T. Polymorphic malware detection using sequence classification methods and ensembles. *EURASIP J. Inf. Secur.* **2017**, *2017*, 2. [CrossRef]
41. Ni, S.; Qian, Q.; Zhang, R. Malware identification using visualization images and deep learning. *Comput. Secur.* **2018**, *77*, 871–885. [CrossRef]
42. Le, Q.; Boydell, O.; Mac Namee, B.; Scanlon, M. Deep learning at the shallow end: Malware classification for non-domain experts. *Digit. Investig.* **2018**, *26*, S118–S126. [CrossRef]
43. 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]
44. Marastoni, N.; Giacobazzi, R.; Preda, M.D. Data augmentation and transfer learning to classify malware images in a deep learning context. *J. Comput. Virol. Hacking Tech.* **2021**, *17*, 279–297. [CrossRef]
45. Darem, A.; Abawajy, J.; Makkar, A.; Alhashmi, A.; Alanazi, S. Visualization and deep-learning-based malware variant detection using OpCode-level features. *Future Gener. Comput. Syst.* **2021**, *125*, 314–323. [CrossRef]
46. Wu, X.; Song, Y.; Hou, X.; Ma, Z.; Chen, C. Deep Learning Model with Sequential Features for Malware Classification. *Appl. Sci.* **2022**, *12*, 9994. [CrossRef]
47. Di Mauro, M.; Galatro, G.; Liotta, A. Experimental review of neural-based approaches for network intrusion management. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 2480–2495. [CrossRef]
48. Dong, S.; Xia, Y.; Peng, T. Network Abnormal Traffic Detection Model Based on Semi-Supervised Deep Reinforcement Learning. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 4197–4212. [CrossRef]
49. Pelletier, C.; Webb, G.I.; Petitjean, F. Deep learning for the classification of Sentinel-2 image time series. In Proceedings of the IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 28 July–2 August 2019.