Alexandria Engineering Journal (2022) 61, 91–99

HOSTED BY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

Malware classification based on double byte feature encoding

Lin Li *, **Ying Ding, Bo Li, Mengqing Qiao, Biao Ye**

School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, Hubei 430065, China
Institute of Big Data Science and Engineering, Wuhan University of Science and Technology, Wuhan, Hubei 430065, China
Hubei Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan, Hubei 430065, China

Received 27 February 2021; revised 12 April 2021; accepted 26 April 2021
Available online 05 June 2021

KEYWORDS

Convolutional Neural Network;
Malware Classification;
Double Byte Feature Encoding;
Feature Selection

Abstract Many researchers analyze malware through static analysis and dynamic analysis technology, and combine it with excellent deep learning algorithm, which has achieved good results in malware classification. However, many researches only use the. ASM file generated by decompiler or. Bytes file represented by hexadecimal for feature extraction. This paper fully integrates the features of these two files, and uses word frequency and two deep learning algorithms to extract 184 opcode features and 16 probability features from ASM file and section file of Kaggle dataset respectively. Then, double byte feature coding method is used to fuse the features of the two files. Finally, convolution neural network is used to classify the fused samples. The experimental results show that the accuracy is 98.68% and the logarithm loss is 0.022.

1. Introduction

Malware can be considered as any software that is deliberately designed to destroy computers, servers or any network. Malicious code is the source of malicious software to achieve its purpose. Malicious code is a kind of program that can destroy computer data, run and destroy the integrity and security of the data by embedding itself into another program without being detected. It has both unauthorized and destructive characteristics. Due to the great harmfulness of malicious code, some corresponding anti-malicious code software and system have been developed to ensure the safety and integrity of the program.

The study of malicious code is one of the research problems in the field of network security. With the development of computer hardware technology, researchers have applied deep learning algorithms to the detection of malicious software. Some outstanding achievements have been made in the fields of natural language processing and image recognition.

Malicious code detection can be divided into static detection and dynamic detection [1]. Combining the two detection methods with the deep learning algorithm is much faster than the traditional malicious code detection method in terms of efficiency. It has been reported that there are many classification techniques based on static and dynamic analysis that can be used to develop effective malware classification systems [2–13].

* Corresponding author.
E-mail address: lilin@wust.edu.cn (L. Li).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

https://doi.org/10.1016/j.aej.2021.04.076

The static analysis methods of malicious code mainly include disassembly, decompiling, string extraction analysis, binary structure analysis and so on. Static analysis usually involves examining an assembly of code without execution. The advantage of static analysis is that it can be analyzed before the malware is executed and the relevant sensitivity of the code can be mined. Researchers can often use deep learning algorithms to classify the malware based on its static characteristics. For example, Jiang et al. [14] converted the assembly instruction set of malicious code into 64bit binary number through vectorization method, and then converted it into images. They classified it through Convolutional Neural Networks (CNN), and achieved 97.87% accuracy on the Big2015 data set released by Microsoft in 2015. Qiao et al. [15] through reverse analysis of malicious code executable file, so as to get the executable assembly code, and then will get assembly instruction as a word, to get to function as a sentence, under this operation, a malicious code is converted to a document, then through Word2Vec algorithm to obtain the word vector for each document, the last of the top 100 training assembly instruction sequences, each document is converted to a matrix, the final accuracy on CNN algorithm is 98.56%. By converting the entire executable file into a sequence, Raff et al. [16] proposed the Malconv model to solve the scaling problem of long sequences, and also took into account the local and global context issues, and the model also had good interpretation ability of analyzing malware annotations. Finally, the model has achieved a very good effect on the classification of malicious samples and benign samples. However, the disadvantage of this method is that it consumes too much GPU resources. Baptista et al. [17] converted malicious software binary into image through malware binary visualization technology, and then analyzed and detected the image through self-incremental neural network, which was able to detect malicious software very efficiently, and this method was a method of unsupervised learning.

Dynamic analysis methods of malicious code mainly include dynamic debugging, snapshot comparison, sandbox, system dynamic behavior monitoring and so on. In dynamic analysis, the behavior of the malware is monitored in real time, so the infected area of the program can often be controlled easily, which is not possible in static analysis. Dynamic analysis can also find out the deformation and confusion of malware more accurately. Li et al [18] extracted the dynamic features of malicious codes by running malicious software under virtual machines to achieve the purpose of protecting physical machines. In this method, the original features of malicious code are filtered and extracted, and the sequence information of malicious code sample call API is obtained. In addition, the traditional N-gram model is improved to make features more effective by adding relevant frequency information and dependencies among APIs. The accuracy of this model is 91.5% under the decision tree J48. Yang [19] divided the four levels of normal layer, sensitive layer, suspicious layer and malicious layer to include all API information, basic behavior and advanced behavior information for rule matching. Adaboost algorithm is used as the main integration framework, and decision tree algorithm is used as the classifier and the final result is obtained. Word2vec model was used to map API into word vector, then extracted word vector features by CNN, and finally classified by Softmax. The effects of these three methods on the classification of dynamic features of malicious software were compared. By encoding three different intrusion detection data sets, Kim et al. [20] coded various dynamic features in intrusion detection into color images and classified them with CNN, with an accuracy rate of close to 89%. Anderson [21] and others by means of dynamic collection target executable instruction track diagram were analyzed, and the figure says markov chain, the vertices are instructions, transition probability estimates by tracking contained in the data, and by using the combination of graphics kernel to create order tracking chart between similarity matrix, by measuring figure on the local and global similarity, in the end, will be sent to the similar matrices support vector machine (SVM) classification. Compared with other signature-based machine learning algorithms, this method has obvious improvement. Ayr[22] proposes an ensemble capsule network model based on the bootstrap aggregating technique. The proposed method is tested on two widely used, highly imbalanced datasets (Malimg and BIG2015). The proposed model achieves the highest F-Score, which is 0.9820, for the BIG2015 dataset and F-Score, which is 0.9661, for the Malimg dataset.

The rest of this article is organized as follows. Section 2 discusses how to extract the characteristics of the ASM file and the byte file. Section 3 describes the double-byte feature encoding technique. Section 4 discusses the classification of malware. Section 5 presents the experimental results and discussion. Section 6 is the conclusion.

2. Feature extraction of ASM files and BYTE files

2.1. Feature extraction of ASM files

ASM file is a kind of file generated by disassembly software. The information of ASM file in the computer is shown in Fig. 1.

The Fig. 1 shows that the ASM file instruction set includes mov, or and CMP operation code, therefore, through statistical opcode appeared in all samples can get all the characteristics that the data set (677), and then separately for each sample, whether the 677 features in the selected 184 features of highest frequency of all the samples as the features of the ASM file. The final extracted Fig. 2 is shown as follows:

2.2. Feature extraction of byte files

The representation of byte files in computers is hexadecimal, and they are stored as shown in Fig. 3:

The byte file is shown in Fig. 3 as a hexadecimal number, so the corresponding sample can be directly converted to a grayscale image. Fig. 4 is the initial grayscale image form of a malware sample:

```
push     eax
     lea      eax, [esp +2CH+var+c]
          move large fs:0, eax
     xor      eax, eax
     push     5      ; MaxCount
```

Fig. 1 ASM file information.

LABEL: 0	
ID: 01kcPWA9K2BOxQeS5Rju	
push: 81	dword: 8
mov: 89	jmp: 12
sub: 5	pusha: 1
lea: 36	shl: 1
……	……
BF: 0	B1: 0

Fig. 2 Individual ASM file characteristics.

```
0040110 5F C3 09 00 …
0040120 45 8B 23 07 …
0040130 03 3F CC CC …
0040140 04 F8 E3 45…
```

Fig. 3 Byte file information.

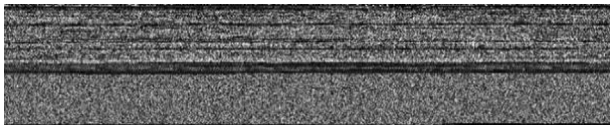

Fig. 4 Initial grayscale picture of a single sample.

The size of the converted grayscale image varies depending on the size of the malware. Therefore, we need to make the size of these pictures consistent. Fig. 5 converts the image from Fig. 4 (enlarged) to a size of 32×32.

2.3. Feature extraction of byte file

For the feature extraction of byte file, CNN and self coding are used to extract them respectively, and the probability that the sample belongs to a certain category is obtained respectively. Each method obtains 8 features, so a total of 16 features can be obtained.

Fig. 5 A fixed size image of a single sample.

2.3.1. Feature extraction based on CNN

After the byte file is converted into a 32×32 fixed size image, it can be extracted by CNN. Fig. 6 shows the architecture of CNN feature extraction.

As shown in Fig. 6, the input is 3232 Gy× image, after five hidden layers and three connected layers, the output is 8, after one softmax, the final output is 8 probability features. Among the five hidden layers, the largest pooling layer in the first four layers is 2×2, and the convolution kernel in the convolution layer is 3×3. After five hidden layers, the input is 3232 Gy× image, and the output size is 2×2×1024. After the first layer of all connected layer, the output is 1000, then through the second layer of all connected layer, the output is 500, and the last time through the all connected layer, the output is 8. After the softmax layer processing, the final result is 8. The related parameters of CNN architecture are shown in Table 1.

When the sample is extracted by the architecture diagram in Fig. 6, eight probability features are generated, and the representation is shown in Fig. 7.

Where id is the name of the sample, labels indicate which category the sample belongs to, and prob0 to prob7 indicate the probability that the sample belongs to a certain sample after being extracted by CNN. For example, the sample named 01kcpwa9k2boxqes5rju on the left is considered to belong to category 0 with a probability of 0.986746013 after feature extraction in the CNN network architecture diagram in Fig. 6, and the total probability of belonging to other categories does not exceed 0.02, while the sample named 1e93cp-p60rhfnit5qfvn on the right belongs to category 0 with a probability of 0.188826576, while the sample named 1e93cp-p60rhfnit5qfvn belongs to category 4 with a probability of 0.811131835 The probability synthesis of other categories is not more than 0.0001. If the probability is high to determine which category the sample belongs to, the sample on the left will be correctly classified as category 0, while the sample on the right will be wrongly classified as category 4. Therefore, for byte files, it is impossible to classify them directly by probability features. However, these features can be regarded as the features of byte file to assist ASM file to classify malicious samples.

2.3.2. Feature extraction based on self-encoding

In order to better assist ASM file features to classify malware samples, on the basis of CNN feature extraction, the byte file features are extracted by self-encoding, so as to increase the proportion of byte file features in the mixed features. Fig. 8 shows the architecture of self = encoding feature extraction.

As can be seen from Fig. 8, after two times of self encoding, the input size of the byte image becomes 32 ×32×256, and then after one time of linear + relu and softmax, the output size is 8, representing the probability of belonging to one of the eight categories. The parameters corresponding to the self-encoding feature extraction architecture are shown in Tables 2, 3 and 4.

As can be seen from Fig. 8 and tables 2, 3 and 4, the final output result is 8, which represents the probability that the sample belongs to 8 categories. Fig. 9 shows the result after the sample is processed by self coding CNN.

The analysis of Fig. 9 is the same as that of Fig. 7. It can be seen that for the sample on the left, the probability of self coding CNN predicting it as class 0 is 0.92691648, which is similar to the prediction of CNN for this sample. The difference is that

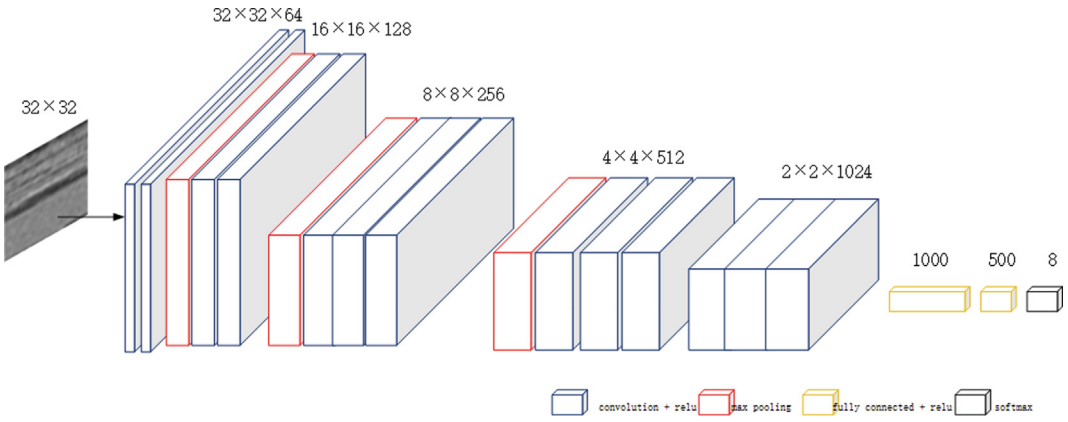

Fig. 6 CNN feature extraction architecture.

Table 1 Parameters of CNN architecture diagram.

Parameters	Value
layers	5
batch_size	100
epochs	5000
learning rate	0.001

Table 2 A architecture parameters of self coding CNN.

parameters	value
weight decay	0.00001
batch_size	100
epochs	100
learning rate	0.001

```
ID: 01kcPWA9K2B0xQeS5Rju          ID: 1E93CpP60RHFNiT5Qfvn
LABELS: 0                         LABELS: 0
prob0: 0.986746013                prob0: 0.188826576
prob1: 0.000028                   prob1: 0.00000292
prob2: 0.000028                   prob2: 0.00000292
prob3: 0.000154652                prob3: 0.00000292
prob4: 0.000028                   prob4: 0.811131835
prob5: 0.000028                   prob5: 0.00000292
prob6: 0.012934998                prob6: 0.000027
prob7: 0.000028                   prob7: 0.00000292
```

Fig. 7 CNN characteristic probability diagram.

for the samples on the right, when they belong to category 0, CNN is more inclined to predict them as category 4, while self coding CNN is more inclined to predict them as category 0, although category 4 accounts for about 26%. Therefore, combining the probability features of the two methods as auxiliary features can improve the accuracy of malware classification.

3. Double byte feature encoding technology

Through sections 2.1 and 2.2 of Chapter 2, 200 mixed features can be obtained from ASM files and byte files in the dataset. The representation of mixed features of a single sample is shown in Fig. 10.

Fig. 10 shows 200 mixed features of a single sample. In the first eight lines, the blue features on the left are eight probabilistic features extracted from the byte file by five layer CNN, while the red ones on the right represent eight probabilistic features extracted from the byte file by encoding, and the light blue features are 184 features extracted from the ASM file by the frequency of features appearing in the sample.

After mixing the features of the samples, it is also necessary to encode them by double byte feature coding technology. As can be seen from Fig. 10, the mixed features of the samples are

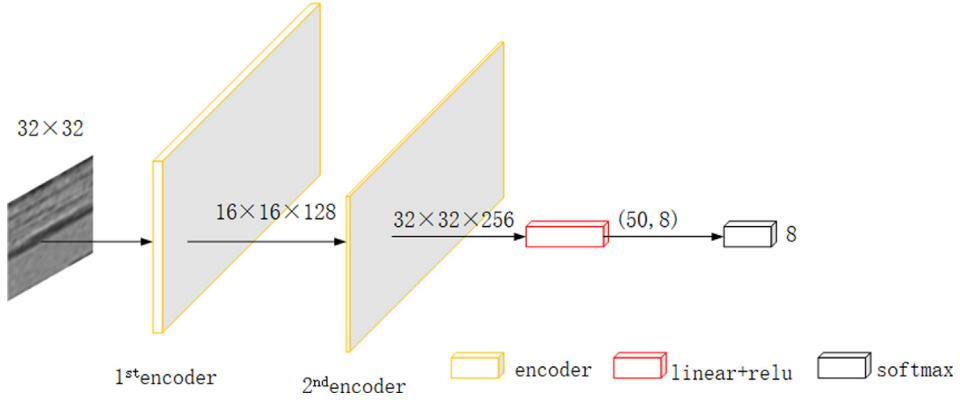

Fig. 8 Architecture of self-encoding feature extraction.

Table 3 The first self-encoding parameters.

	layers	input	kernal	filters	strides	padding	output
Enc	conv	32,321××	3×3	128	1	1	3,232,128××
ode	relu	3,232,128××					3,232,128××
	pool	3,232,128××	2×2		2		1,616,128××
Dec	conv	1,616,128××	2×2	1	2		32,321××
ode	relu	32,321××					32,321××

Table 4 The second self coding parameters.

	layers	input	kernal	filters	strides	padding	output
Enc	conv	1,616,128××	3×3	128	1	1	1,616,256××
ode	relu	1,616,256××					1,616,256××
	pool	1,616,256××	2×2		2		88,256××
Dec	conv	88,256××	2×2	128	2		1,616,128××
ode	relu	16,161××					1,616,128××

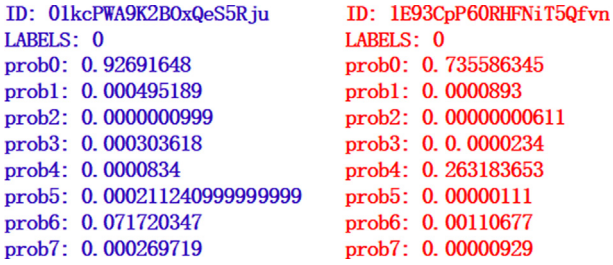

Fig. 9 Self-encoding feature probability diagram.

all numerical (integer and decimal), and the integer features can be normalized to [0,1] After all features are converted to decimals, a single numeric feature can be converted to two bytes by formula (2). Formula (1) and formula (2) are as follows.

$$x_{norValue} = \frac{x_{\max Value} - x_{originalValue}}{x_{\max Value} - x_{\min Value}} \tag{1}$$

In formula (1), $x_{\max Value}$represents the maximum value of the feature in the dataset, $x_{\min Value}$ represents the minimum value of the feature in the dataset, $x_{originalValue}$ represents the value of a sample of the feature, and $x_{norValue}$ represents the normalized value of the sample.

$$hex = \begin{cases} 0X0000 & nor = 0 \\ hexVal(nor)[0:4] & 0 < nor < 1 \\ 0XFFFF & nor = 1 \end{cases} \tag{2}$$

Formula (2) means that the normalized value is converted to hexadecimal decimal, and on this basis, the first four digits are intercepted to form a two-byte feature.

Through formula (1) and formula (2), the features in Fig. 10 can be transformed into a gray image with a size of 20. The converted image is shown in Fig. 11.

The eight gray-scale images (enlarged) shown in Fig. 11 are composed of one image randomly selected from eight categories after double byte feature encoding.

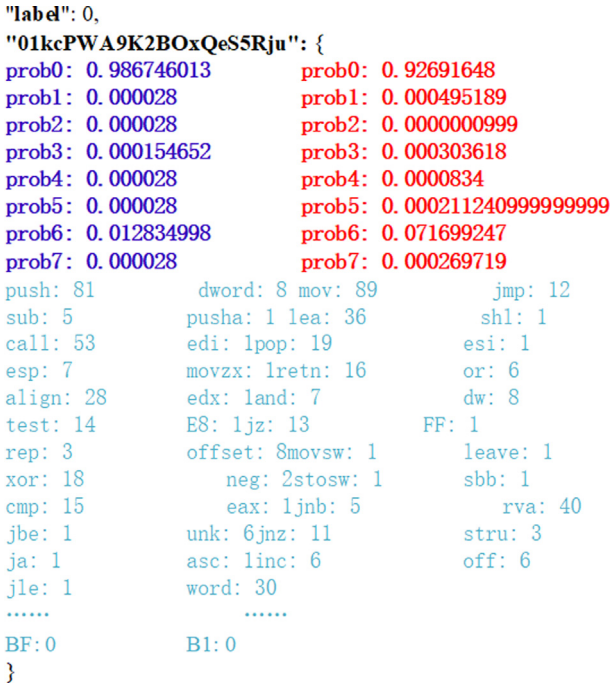

Fig. 10 Mixed characteristics of a single sample.

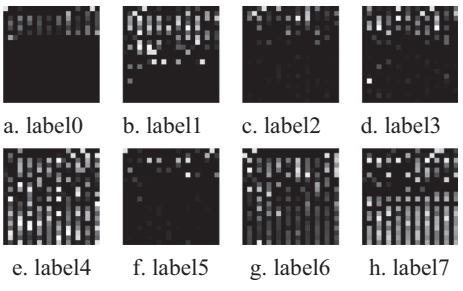

Fig. 11 Mixed feature gray image.

4. Classification of malware

4.1. Datasets

In this paper, the dataset used for malware classification is Big2015 released by Microsoft and on the platform of kaggle [25]. The size of the dataset after decompression is 500G. At present, the datasets have become the benchmark dataset for malware researchers, and a large number of references have been made to it. Each sample in the datasets contains two kinds of files, one is the .ASM file generated by IDA tool disassembly, and the other is the .Byte file in hexadecimal.

Big2015 data set contains a total of 21,741 samples, of which 10,868 samples are used as training sets and have been labeled. The sample distribution is shown in Fig. 12.

As can be seen from Fig. 12, kelihos with label 2 has the largest number of samples among the nine categories_ For ver3, the number of samples is 2942, while Simda with label 4 has the least number of samples, only 42. Therefore, the fourth-class samples are removed, and the remaining samples are divided into training sets and test sets in a way close to 3:1.

4.2. Malware classification

After getting the gray image formed by the mixed features of the samples in Section 3, the next step is to classify the mixed sample datasets. Before introducing the classification of the mixed sample data set, first introduce the whole process of the experiment. The experimental process is shown in Fig. 13.

The specific flow of the left part has been described in Section 2 and section 3, while the detailed flow of the right part from mixed data set to malware detection result will be described in Fig. 14.

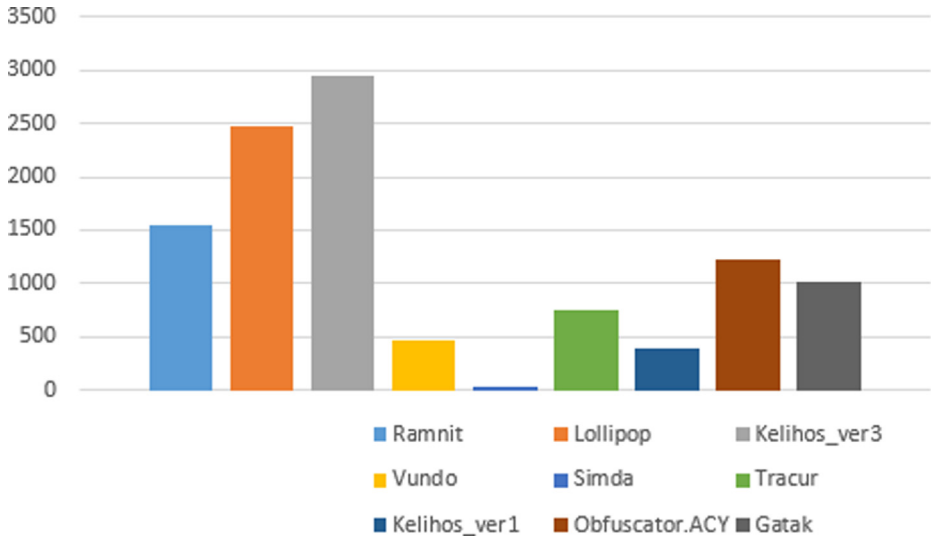

Fig. 12 Distribution of datasets.

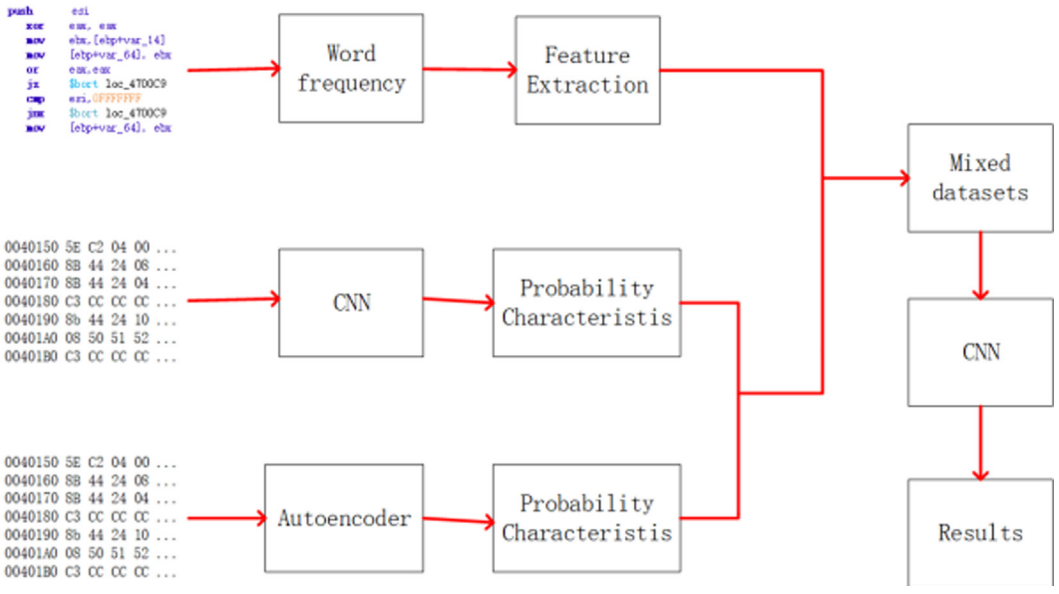

Fig. 13 Overall process architecture.

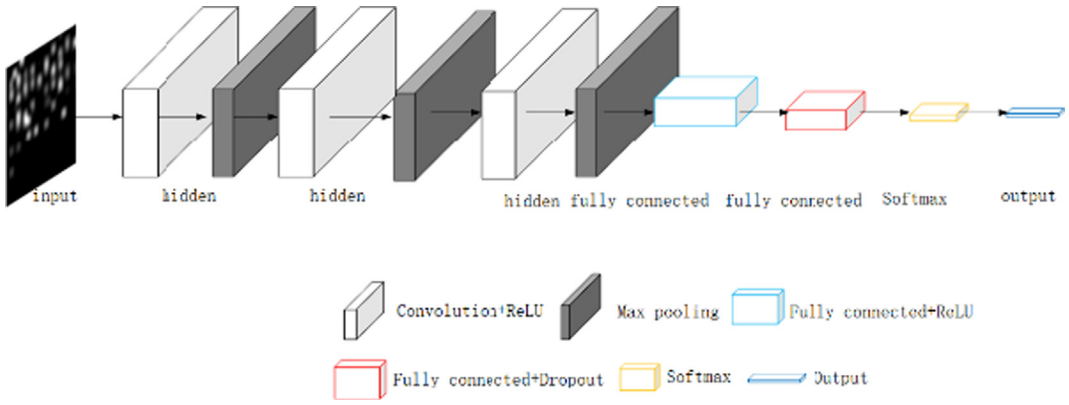

Fig. 14 Malicious code classification architecture.

As can be seen from Fig. 14, the gray image formed by mixing features is the input layer. After three times of hiding layer and two times of all connected layer operation, the output layer is finally obtained by softmax, and the final result of the output layer is 8.

5. Experimental results and discussion

5.1. Evaluation index

In this paper, accuracy and log loss are used to show the advantages of the proposed double byte feature coding method, in which accuracy means the number of all samples with correct prediction divided by the number of total samples, and the calculation of log loss is shown in formula 3.

$$\text{logloss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}) \tag{3}$$

In Formula 3, N is the total number of samples, M is the number of categories of samples, y is the real label of samples, p is the probability of predicted samples, and log is the natural logarithm.

5.2. Experimental results and comparative analysis

2192 samples are randomly selected from the data set as the training sets. The number of samples in each category is shown in Table 5.

The prediction labels of all samples in the test sets are shown in Table 6.

Row i nd column j in Table 6 indicate the probability that the sample originally belonging to class I will be predicted as the j-th sample. For example, the diagonal is the sample with correct prediction, and the fourth column in row 0 indicates that two samples belonging to category 0 are misjudged as category 4, and so on.

Table 7 shows the accuracy of ASM data set, byte data set and mixed data set on eight malicious families.

Table 5 Sample number of various classes in test set.

Class	Numbers	Labels
Ramnit	299	0
Lollipop	523	1
Kelihos_ver3	587	2
Vundo	105	3
Tracur	159	4
Kelihos_ver1	59	5
Obfuscator.ACY	262	6
Gatak	198	7

Table 7 Accuracy of ASM, byte and mixed data sets.

Labels	ASM data sets accuracy	byte data sets accuracy	mixed data sets accuracy
Ramnit (0)	91.20%	94.88%	98.66%
Lollipop (1)	90.60%	95.53%	99.24%
Kelihos_ver3 (2)	89.80%	93.92%	100%
Vundo (3)	92.90%	95.45%	100%
Tracur (4)	91.75%	95.75%	95.60%
Kelihos_ver1 (5)	91.60%	95.82%	100%
Obfuscator.ACY (6)	92.20%	95.83%	96.56%
Gatak (7)	93.10%	96.89%	97.47%

Table 6 Malicious code family classification confusion matrix.

Labels	0	1	2	3	4	5	6	7
0	295	0	0	0	2	1	1	0
1	0	519	0	0	0	0	4	0
2	0	0	587	0	0	0	0	0
3	0	0	0	105	0	0	0	0
4	2	0	0	0	152	0	5	0
5	0	0	0	0	0	59	0	0
6	8	0	0	1	0	0	253	0
7	1	2	1	0	0	0	1	193

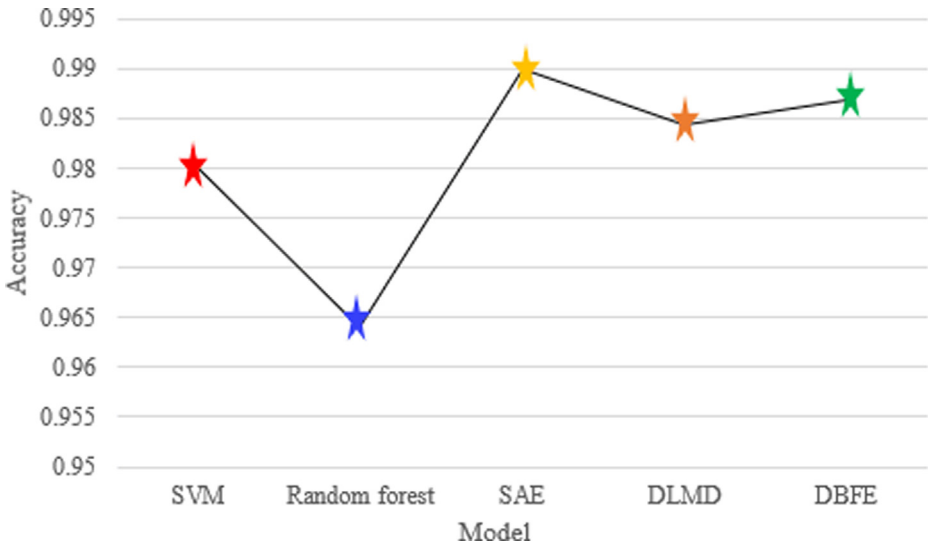

Fig. 15 Accuracy comparison chart of different methods.

According to Table 7, for the method using ASM dataset [23] only, the classification accuracy of ASM dataset in malicious families is not more than 94%, and the classification accuracy rate in all families is lower than that of byte data set only [24]. In the accuracy rate of each kind of mixed data set, except the accuracy rate of the fourth kind is 95.6%, which is slightly lower than that of the byte data set, the accuracy rate of the other seven kinds of mixed data sets is better than that of the single ASM data set and the byte data set.

Compared with the combination of other methods and mixed data sets, the accuracy and loss of the whole sample are shown in Fig. 15 and Table 8 respectively.

As can be seen from Fig. 15, five different methods are used to conduct comparative experiments on mixed data sets, the lowest accuracy rate was random forest, which was only 96.41%. The accuracy of the proposed method is 98.68%, compared with SVM and DLMD, the accuracy is 0.68% and 0.25% higher respectively. Then compare the log loss of different models.

It can be seen from table 8 that the log loss of the proposed method is the smallest, only 0.022, the second is DLMD method, and its loss value is only 0.0763, The SAE method with the highest accuracy has a log loss of 0.227, 0.205 higher than DBFE's log loss.

In conclusion, the double byte feature coding method proposed in this paper has the lowest log loss in the case of good accuracy performance for malware mixed data sets, and it is an excellent method for malware classification.

Table 8 Comparison of log loss in different models.

Model	Log loss
SVM	0.872
Random forest	0.9036
SAE	0.227
DLMD	0.0763
DBFE	0.022

6. Conclusion

In this paper, a double byte feature encoding method is proposed, and .ASM file,.Byte file are used to classify malware datasets at the same time. The experimental results show that the accuracy and log loss of the double-byte feature encoding method and the method of using both .ASM file and .Byte file are better than those of word2vec and the method of using only a single datasets.

The double byte feature encoding method still has this limitation, for example, the requirement for features and the number of samples are relatively high, so the future work will try to increase the number of samples and use natural language processing algorithms to process features; At the same time, the new deep learning algorithm will be combined to simplify the process.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research work was supported by Youth Research project of Hubei Provincial Department of Education(20Q022), National College Students' Innovation and Entrepreneurship Training Program (201810488012) and the National Natural Science Foundation of China(U1803262, 61572381).

References

[1] Hyunjae Kang, Jae-wook Jang, Aziz Mohaisen, Huy Kang Kim, Detecting and Classifying Android Malware Using Static Analysis along with Creator Information, Int. J. Distrib. Sens. Netw. 2015 (2015) 1–9.

[2] Wadi' Hijawi, Ja'far Alqatawna, Ala' M. Al-Zoubi, Mohammad A. HassonahaHossamFaris, Android botnet detection using

machine learning, Ingénierie des Systèmes d'Information 25 (1) (2020) 127–130.
[3] B.Y. Ouyang, X. Liu, C. Xu, Malware behavior evaluation system based on support vector machine, Computer applications 4 (2015) 972–976.
[4] C. Jiang, Y.P. Hu, K. Si, et al, Malicious file detection method based on image texture and convolutional neural network, Computer applications 038 (010) (2018) 2929–2933.
[5] G.H. Liao, J.Y. Liu, Malicious code detection method based on data mining and machine learning, Information security research 2 (1) (2016) 74–79.
[6] B. Lainjo, Network security and its implications on program management, International Journal of Safety and Security Engineering 10 (6) (2020) 739–746.
[7] G. Wang, R.R. Hao, S.W. Gao, Malicious code detection method based on wig-ga feature selection algorithm, Computer science and applications 8 (3) (2018) 266–274.
[8] A.C. Metlapalli, T. Muthusamy, B.P. Battula, Classification of social media text spam using VAE-CNN and LSTM model, Ingénierie des Systèmes d'Information 25 (6) (2020) 747–753.
[9] George E. Dahl; Jack W. Stokes; Li Deng; Dong Yu. LARGE-SCALE MALWARE CLASSIFICATION USING RANDOM PROJECTIONS AND NEURAL NETWORKS. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2013: 3422-3436.
[10] V.A. Benjamin, H. Chen, Machine Learning for Attack Vector Identification in Malicious Source Code, Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on IEEE (2013) 21–23.
[11] V. Sstla, V.K.K. Kolli, L.K. Voggu, R. Bhavanam, S. Vallabhasoyula, Predictive model for network intrusion detection system using deep learning, Revue d'Intelligence Artificielle 34 (3) (2020) 323–330.
[12] Joshua Saxe, Richard Harang, Cody Wild, Hillary Sanders, A Deep Learning Approach to Fast, Format-Agnostic Detection of Malicious Web Content. 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA 2018 (2018) 8–14.
[13] A. Al-Dujaili, A. Huang, E. Hemberg, U.M. O'Reilly, Adversarial deep learning for robust detection of binary encoded malware. Proc. - 2018 IEEE Symp, Secur. Priv. Work. SPW (2018) 76–82.
[14] Y.K. Jiang, Y. Wu, F.T. Zou, Malicious code classification model based on image vector, Communication technology 51 (12) (2018) 2953–2959.
[15] Y.C. Qiao, Q.S. Jiang, L. Gu, X.M. Wu, Research on malicious code classification based on assembly instruction word vector and convolutional neural network, Information network security 04 (2019) 20–28.
[16] Edward Raff,Jon Barker, Jared Sylvester,Robert Brandon, Bryan Catanzaro,Charles Nicholas. Malware Detection by Eating a Whole EXE. Thirty-Second AAAI Conference on Artificial Intelligence. 2018: 1-13.
[17] Baptista, I., Shiaeles, S., Kolokotronis, N. A Novel Malware Detection System Based on Machine Learning and Binary Visualization. 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai China, 2019: 1-6.
[18] Li Meng, Jia Xiaoqi, Wang Rui, Lin Dongdai, A malicious code feature selection and modeling method, Computer applications and software 08 (2015) 272–277.
[19] Yang Ye, Research on behavior based malicious code detection, Xi'an University of Electronic Science and technology, 2015.
[20] Taejoon Kim, Sang C. Suh, Hyunjoo Kim, Jonghyun Kim, Jinoh Kim, An Encoding Technique for CNN-based Network Anomaly Detection, 2018 IEEE International Conference on Big Data (Big Data). Seattle, WA, USA (2018), 29602965.
[21] Blake Anderson, Daniel Quist, Joshua Neil, Curtis Storlie, Terran Lane, Graph-based malware detection using dynamic analysis, J. Comput. Virol. 7 (4) (2011) 247–258.
[22] A. Ayr, U. Nal, H. Da, Random CapsNet forest model for imbalanced malware type classification task, Computers & Security (2021) 102–133.
[23] C.H. Lo, T.C. Liu, I.H. Liu, et al, Malware Classification using Deep Learning, Proceedings of International Conference on Artificial Life and Robotics 25 (2020) 126–129.
[24] W. Jun-Ling, W. Shuo-Hao, Malicious Classification Based on Deep Learning and Visualization, 2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD) (2019) 223–228.
[25] Kaggle. Microsoft malware classification challenge (BIG 2015). [2018-05-07]. https:// kaggle.Com/c/malware classification/.