FCNNMD: A NOVEL FUSION METHOD BASED ON CONVOLUTIONAL NEURAL NETWORK FOR MALWARE DETECTION

Jing Zhang[1] and Yu Wen[2]

[1]Institute of Information Engineering, CAS & School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[2]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

ABSTRACT

Malicious software are rampant and do great harm. The present mainstream malware detection technology has many disadvantages, such as high labour cost, large system overhead, and inability to detect new malware. We propose a novel fusion method based on convolutional neural network for malware detection (FCNNMD). For the sample imbalance problem faced by the convolutional neural network malware detection method, the non-malicious sample is added by means of generating anti-network generation, etc., to achieve the same number as the malicious sample. For the problem of low accuracy of single model detection, high false positive rate and false negative rate, a malware detection model is constructed by means of model fusion. The model combines four classical convolutional neural network structures. Experiments show that this method can effectively improve the accuracy and robustness of the model. Our method does not need actual running software and has high detection efficiency.

KEYWORDS

Malware Detection, Grayscale Image, Convolutional Neural Networks, Model integration

1. INTRODUCTION

In recent years, the Internet security reports of major security manufacturers at home and abroad show that viruses, worms, trojans and other malicious software are ram-pant and do great harm. In addition, the number of malware and variant malware is still growing rapidly, and the network security situation is still very serious. The present mainstream malware detection technology has many disadvantages, such as high labour cost, large system overhead, easy to be bypassed by malware, and inability to detect new malware.

We propose a novel fusion method based on convolutional neural network for malware detection (FCNNMD). The basic idea is to use the grayscale image generation algorithm to convert the executable file into grayscale images firstly, and then use the convolutional neural network to build a grayscale image classification model. By classifying grayscale images, the purpose of detecting malware is achieved indirectly.

The detection method of malware based on convolutional neural network cannot only reduce human cost and system overhead, improve detection efficiency, but also detect new types of

David C. Wyld et al. (Eds): SPTM, IPPR, AIS, CSIT, DaMi, AMLA - 2020
pp. 01-16, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.100601

malware. In addition, the detection method of malware based on convolutional neural network does not need to rely on artificial features. The main work of this thesis includes:

- For the sample imbalance problem faced by the convolutional neural network malware detection method, the non-malicious sample is added by means of generating anti-network generation, etc., to achieve the same number as the malicious sample. Experiments show that this method can effectively improve the accuracy of the model, reduce the false positive rate and false negative rate of the model.

- For the problem of low accuracy of single model detection, high false positive rate and false negative rate, a malware detection model is constructed by means of model fusion. The model combines four classical convolutional neural network structures. Compared with the single model detection, the accuracy of the model is further improved, and the false alarm rate and the false negative rate are further reduced.

- Aiming at the shortcomings of high complexity and low detection efficiency of model fusion method, a detection model with low complexity and high detection efficiency is designed and implemented. The accuracy, false positive rate and false negative rate of the model are passed. The model obtained by the model fusion method is basically the same, but the detection efficiency is improved by 3.15 times.

The malware detection method proposed in this paper can achieve an accuracy rate of about 97%, which is about 18% higher than the detection method based on texture fingerprint for malicious code variants. The detection method of malware based on convolutional neural network does not need actual running software, has high detection efficiency, does not rely on the construction of feature database, etc., and can greatly reduce labour cost and maintenance cost.

The rest of this paper is organized as follows: we briefly review the related work in Section 2, and we describe the proposed method in Section 3. The experimental results and discussion are presented in Section 4. Finally, we give conclusion and future work in Section 5.

2. RELATED WORK

In essence, malware detection technology is divided into two categories: static analysis and dynamic analysis. At present, most security manufacturers still rely on signature for malware detection, which belongs to static analysis [1]. This method can efficiently detect known malware, but it cannot detect new malware [2-4]. Dynamic analysis [5] is mainly based on the behaviour of software to detect malware. It usually need to actually run software to determine whether a software is malware, the most common practice is to rely on sandboxing programs. This detection method [6] has the disadvantages of high overhead, easy to be discovered by the defence detection mechanism of malware, and easy to be bypassed by malware.

Detection techniques based on machine learning and deep learning can detect new malware [34-36]. The detection method based on machine learning [7-9] mainly starts with the text structure of malware and extracts artificial features from many angles. [10] mainly identified malware by clustering, it first analysed the API of a large number of malware calls and used these API to form the dataset, then it extracted the subject from this dataset to cluster, and identified malware similar to known malware in the dataset. This method is obviously powerless against new malware [16]. Hyrum S et al. constructed an open source, large-scale PE format file data set, modeled and analysed the data set with the methods of machine learning and deep learning. The accuracy is about 95% [11]. The accuracy of detection method based on machine learning is

usually low, and the number of samples used is very small, mostly between hundreds and thousands, and the robustness of the model is not high.

Deep learning [18] outperforms machine learning in many fields. In recent years, many researchers at home and abroad have shifted their research focus to malware detection based on deep learning [12-14]. Most of these methods are based on convolutional neural network [19-21] and recurrent neural network. Among them, the detection methods based on recurrent neural network structure are mostly based on software API call sequences to construct data sets. The accuracy of this kind of detection method is about 97%, but this kind of detection method has the problem of low detection efficiency, at the same time, because the sample is difficult to collect, most of them have the problem of small number of samples and low robustness of the model. The approach of [15] is to obtain the API call sequence of the software and consider the return value of each API. Finally, the classification model is established using the recurrent neural network. In [16], it obtains the API call sequence of the software and the API call sequence of the C language library, and then uses the recurrent neural network and echo state network to establish the classification model. The approach of [33] converts binary executable file into grayscale image, then extracts texture fingerprint based on grayscale image, it proposes a malicious code variant detection method based on texture fingerprint. Edward Raff et al. use convolutional neural network model that based on the bytecode sequence of the whole executable file, and a relatively high detection rate is also obtained, but the main problem is that the hardware requirements are too high, the computation is very large, and the model is relatively simple [17].

We propose a novel fusion method based on convolutional neural network for malware detection (FCNNMD). This method does not need to construct the malicious code feature library, but trains a grayscale image classification model through convolution neural network, which greatly reduces the labour cost and maintenance cost. Importantly, our model can detect new viruses. Compared with other methods of using machine learning to detect malware, the method does not need to rely on artificial features, but rely on convolutional neural network to automatically slave image species to extract features. This method does not need the actual running software, the detection efficiency is high, and the accuracy rate is about 97%. At the same time, samples are relatively easy to collect. This method can improve the accuracy and robustness of the model based on large sample training model.

3. PROPOSED METHODOLOGY

The overall process of our method consists of two parts: the classification part and the detection part (see Figure 1). First, we need to collect a large number of executable files containing malicious programs and a large number of executable files that do not contain malicious programs to build an executable file set. Then all files in the executable file set are converted to grayscale images using grayscale image generation algorithm to get a grayscale image set, and then we use convolutional neural network to train a classification model based on this grayscale image set, which can then be used in the detection process to detect whether an executable file contains malicious programs.

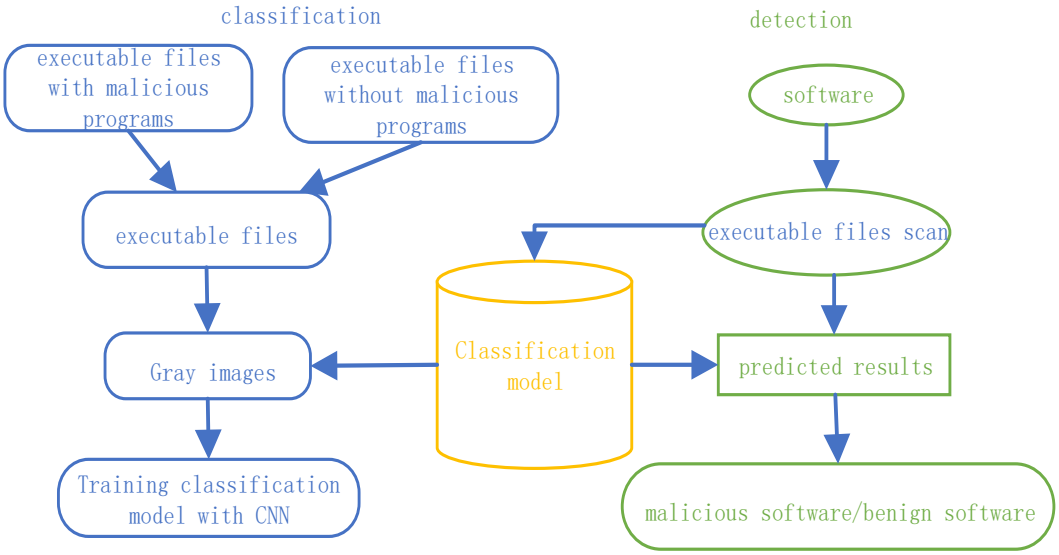


Figure 1. The overall flow of malware detection based on convolutional neural networks

3.1 Grayscale Image Conversion

Among the malware detection methods based on convolutional neural network, the executable file should be converted to grayscale image first, then the malware is detected based on the texture features in the grayscale image. B2M grayscale image generation algorithm is usually used when converting an executable file to a grayscale image [33]. The algorithm is to read any binary file every 8 bits as an unsigned integer (range between 0~255), and then set a fixed row width directly according to experience, so that after the whole file reading is finished, a two-dimensional array of unsigned integers will be obtained. Grayscale images of executable files containing the same family of malicious programs have similar texture features. Grayscale images of different executable files contained in the same software have similar texture features. The previous work does not specify the height and width of the grayscale image. This paper proposes that the form of the grayscale image should meet the following two conditions:

● The width of grayscale images should not be greater than the input width of convolutional neural networks. This ensures continuity of the header information and information of the executable file as much as possible while training the model. The header of the executable file is a key part of judging whether an executable file contains malicious programs. Experiments show that the accuracy can be increased about 2 percentage points by ensuring that the header information of the executable file is not lost.

● Grayscale images corresponding to executable files of similar size should also have similar image sizes. Executable files containing the same family of malicious programs are usually similar in size, because these executable files have similar functions, there is a great similarity between files. When converting these executables into grayscale images, it is ensured that the grayscale images have similar sizes, which ensures that the corresponding grayscale images of these executables have similar textures.

3.2 Sample Imbalance

In order to solve the problem of sample imbalance, after we get the grayscale image set, we adopt the following two ways to increase the number of grayscale image corresponding to the executable file without malicious program in the grayscale image set:

By cutting, flipping, flipping, etc. It is a more common method of data enhancement and can play a role on most data sets;

We use the grayscale image corresponding to the executable file without malicious program to train a generative adversarial network [32] model that can generate the grayscale image corresponding to the executable file without malicious program. First, we create a set of executable files that do not contain malicious programs, and convert all executable files into grayscale images to get a grayscale image set, then we select the image from the grayscale image set and input it into discriminator to let the discriminator judge whether the picture is generated by the generator. In addition, we randomly generate a set of noise data and input noise data into the generator to generate the image, then we input generated image into the discriminator to let the discriminator judge whether the image is generated by the generator. Finally the model adjusts the parameters in the generator and discriminator according to the judgment result of the discriminator, and continuously improve the generator's generating ability and discriminator's discriminant ability. After training, we use the generator to generate images to add to the dataset. Figure 2 shows the images of generator generated.

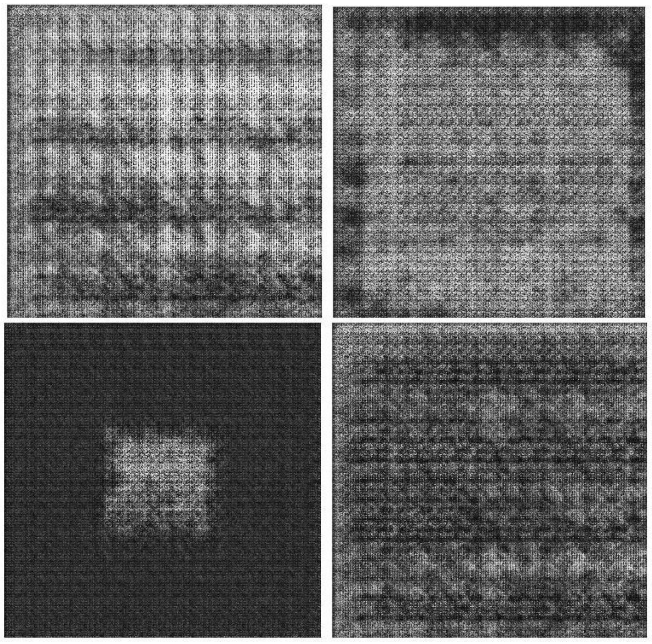


Figure 2. The images of generator generated

The structure diagram of the generator and discriminator in the generative adversarial network is shown in Figure 3.

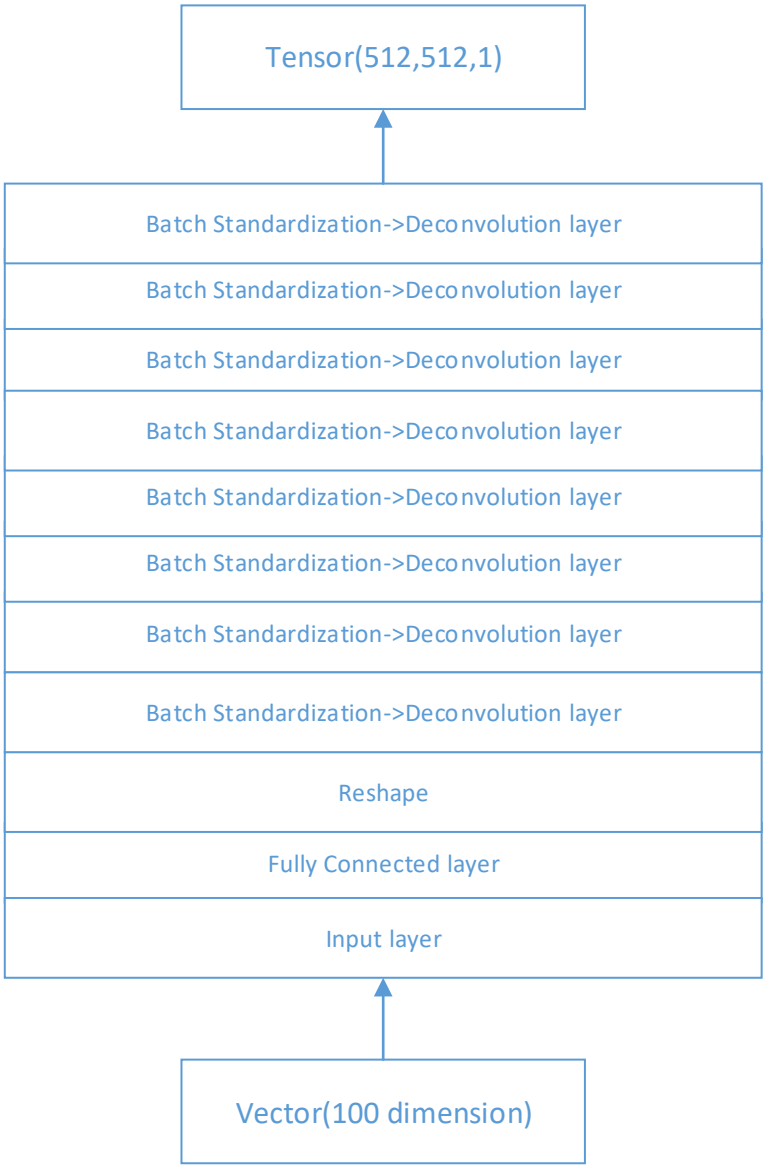

Tensor(512,512,1)
Batch Standardization->Deconvolution layer
Batch Standardization->Deconvolution layer
Batch Standardization->Deconvolution layer
Batch Standardization->Deconvolution layer
Batch Standardization->Deconvolution layer
Batch Standardization->Deconvolution layer
Batch Standardization->Deconvolution layer
Batch Standardization->Deconvolution layer
Reshape
Fully Connected layer
Input layer
Vector(100 dimension)

Figure 3. Generator structure diagram (a) Discriminator structure diagram (b)

Through the above two ways, we increase the original grayscale image set without malicious program executable corresponding to the number of grayscale image set to get the new grayscale image set, we delete part that contains malicious program executable file corresponding to the grayscale image, so that the number of two kinds of images in the data set is the same.

3.3 Model fusion

The model based on VGG16 [25], Inception-v3 [26-29], Xception [31] and ResNet50 [22] [30] is analyzed in detail by experiments (section 4). The following conclusions were drawn: VGG16 can effectively improve the accuracy of the model and reduce the false alarm rate; deep separable convolution layer in Xception can effectively reduce the failure rate of the model; the model with the least time overhead is the Incepiton-v3 model and its detection efficiency is highest. Based on

the above conclusions, this paper designs and implements a complete model from bottom up, as shown in Figure 4.

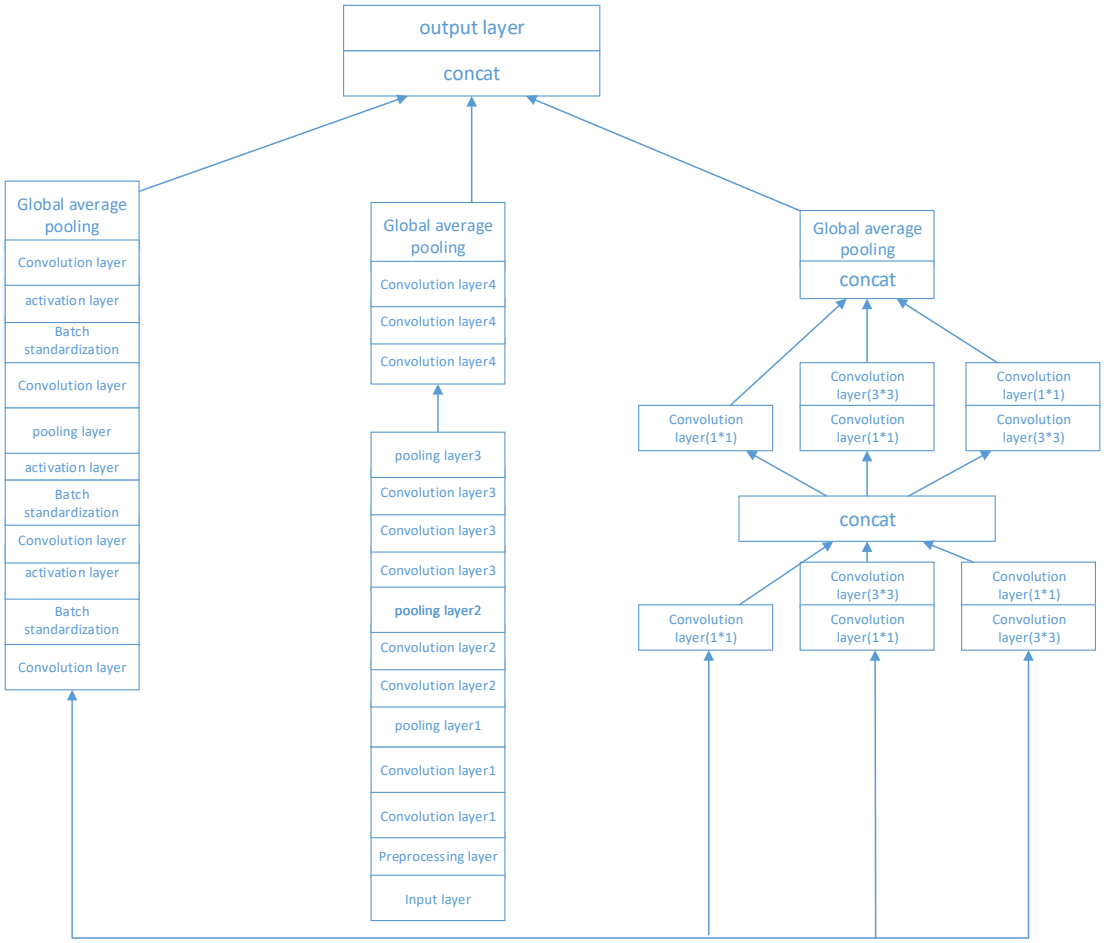


Figure 4. Custom model structure diagram

The input of the model is a grayscale picture, and a preprocessing layer is added after the input layer. The preprocessing layer is responsible for filling the grayscale picture with height or width less than 512 pixels, while scaling all the pixel values in the picture to between 0 and 1.

The main part of the model is mainly composed of 4 convolution pooling modules, each convolution pooling module is composed of multiple convolution layers and 1 pooling layer, the size of the convolution kernel of all convolution layers is 3*3, the step size is 1 and the padding mode is same padding. This part mainly takes into account that the accuracy and false positives of VGG16 in all single models are the lowest, so the trunk part adopts a structure similar to the VGG16 to ensure that the model has high accuracy and low false alarm rate.

The left branch of the model is mainly composed of 4 deep separable convolution layers, it mainly takes into account that the missing rate of Xception in all single models is the lowest, and then it is speculated that the Xception deep separable convolution structure has a role in reducing the missing rate of the model. At the same time, to reduce the complexity of the model, multiple deep separable convolution layers are used as branches to further extract features based on the first three convolution pooling modules of the trunk part, it is expected to reduce the missing rate

of the model. Finally, the batch standardization layer is added to the left branch to accelerate the model convergence and prevent overfitting.

The right branch of the model contains multiple branch structures, it mainly draw lessons from the idea of Inception Module. First, 1*1 convolution is used in multiple branches .1*1 convolution can organize information across channels. Secondly, after each 1*1 convolution operation, the convolution results are nonlinearly calculated using the ReLU activation function. More nonlinear operations are introduced for the model to enhance the fitting ability of the model. Using this structure can also further increase the width of the model and the adaptability to the scale of the network, so there is multi-scale information inside. Moreover, the efficiency of this structure is relatively high. The whole right side branch of the model is mainly responsible for extracting more features in a more efficient way to further improve the performance of the model.

Finally, both the trunk part and the left and right side branches obtain a one-dimensional vector through a global average pooling layer. These three one-dimensional vectors are connected, then the grayscale image through the output layer is the probability of the grayscale image corresponding to the executable file containing the malicious program.

4. EXPERIMENTS

4.1. Experimental Settings

4.1.1. Datasets

We collect a large number of executable files containing malicious programs and executable files without malicious programs through various channels. Executable files that do not contain malicious programs are mainly from the mainstream Windows operating system, and executable files that contain malicious programs are mainly from sites that specifically collect malware. The composition of the data sets is shown in Figure 5:

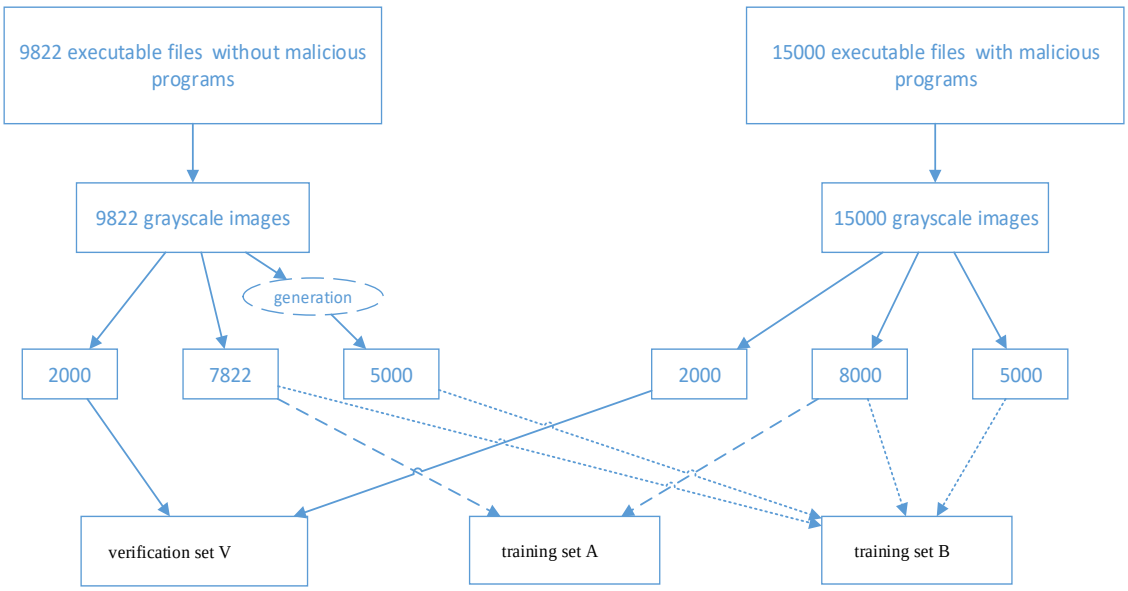


Figure 5. The division of data sets

Figure 5 shows that a total of 9822 executable files without malicious programs and 15000 executable files containing malicious programs participated in the construction of the data set. First, all the executable files are transformed into grayscale images using the improved B2M grayscale image generation algorithm. Then, based on 9822 grayscale images corresponding to executable files that do not contain malicious programs, 5000 grayscale images are generated using the method mentioned above. Then 9822 grayscale images of executable files that do not contain malicious programs were divided into two parts, one containing 2000 and the other containing 7822. Meanwhile, 15000 grayscale images of executable files containing malicious programs are divided into 3 parts. Finally, the verification set V, training set A and training set B are constructed according to the graph.

4.1.2. Metrics

For evaluating the performance of each model on the verification set V, the main evaluation indexes are accuracy, false alarm rate, missing report rate and time overhead. Time overhead is the time taken by each model to evaluate 4000 grayscale images V the validation set.
Accuracy:

$$Acc = \frac{TP + TN}{TP + TN + FN + FP} \times 100\% \tag{1}$$

False alarm rate:

$$False\ alarm\ rate = \frac{FN}{TP + FN} \times 100\% \tag{2}$$

Missing report rate:

$$Missing\ report\ rate = \frac{FP}{FP + TN} \times 100\% \tag{3}$$

4.1.3. Detailed Implementation

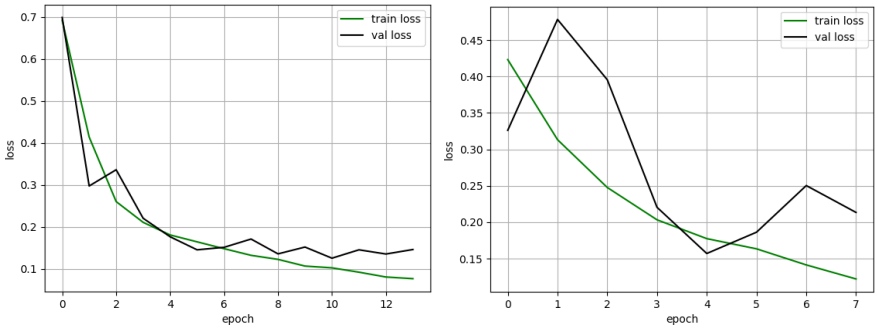
In this paper, the operating system used in the experiment is Linux system (Ubuntu 16.04), the width and height of grayscale images are defined as 512 pixels. DCGAN [32] is adopted in this paper. In our model, batch size is 8, epoch number is 20, and Adam is the optimization algorithm used. In order to prevent overfitting, the early stop strategy is also used.

4.2. Performance Comparison

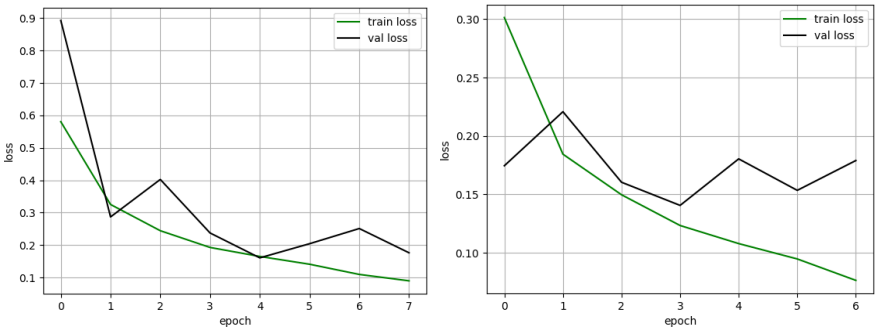
4.2.1. Single Model Comparison

VGG, Inception, ResNet, Xception are four classical convolutional neural network structures. Most of the convolutional neural network structures are based on the further combination and improvement of these network structures, so in this paper, we first select these four classical convolutional neural networks to evaluate the performance, then we try to construct a better model based on the performance of these four convolutional neural networks.

We have modified the network structure by adding a preprocessing layer before the input layer, which will fill images with height and width less than 512 pixels in the form of nearest neighbor interpolation. The preprocessing layer will also scale all pixel values of images to between 0 and 1, which can accelerate model convergence.

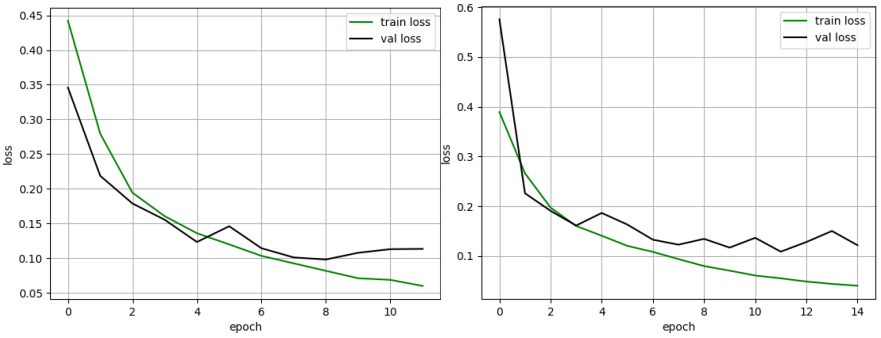


(a) VGG (b) Inception

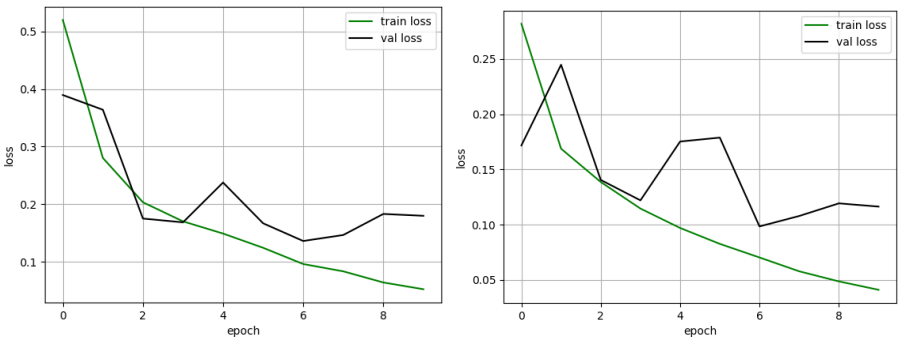


(c) ResNet (d) Xception

Figure 6. Loss curve (training set A, verification set V)



(a) VGG (b) Inception



(c) ResNet (d) Xception

Figure 7. Loss curve (training set B, verification set V)

Table 1. The evaluation result of each model on training set A and verification set V

Model	accuracy	false alarm rate	missing report rate	time overhead
VGG16	95.92%	3.46%	4.59%	58s
ResNet50	94.40%	6.26%	4.94%	54s
Xception	95.16%	6.87%	2.80%	62s
Inception-v3	94.20%	5.95%	5.65%	47s
average	94.92%	5.64%	4.50%	/

Table 2. The evaluation result of each model on training set B and verification set V

Model	accuracy	false alarm rate	missing report rate	time overhead
VGG16	96.85%	2.80%	3.47%	58s
ResNet50	95.34%	5.02%	4.30%	54s
Xception	96.64%	3.97%	2.75%	62s
Inception-v3	96.30%	3.80%	3.42%	47s
average	96.28%	3.90%	3.49%	/

The following conclusions can be drawn by observing the above figure and tables:

● The evaluation results of the four models on the validation set V using the training set B are better than that using the training set A. This proves that the scheme used in this paper can improve the model performance and enhance the generalization ability of the model. In order to solve the problem of sample imbalance, we use the generative adversarial network generation network as one of the ways to expand the data set, and the generative adversarial network generates samples with similar distribution but not exactly the same as the original sample, so we can consider using the generative adversarial network to further expand the data set.

● For VGG16, the false alarm rate is the lowest and the accuracy is the highest, which shows that the VGG16 structure can effectively improve the accuracy of the model and reduce the false alarm rate of the model;

● Whether using training set A or training set B training model, the underreporting rate of Xception validation set is obviously lower than that of other models, which indicates that the deep separable convolution layer in Xception can effectively reduce the missing rate of the model.

● The model with the least time overhead is the Incepiton-v3 model, which shows that the Inception-v3 model is the most efficient.

4.2.2. Fusion Model Comparison

First, we design a simple model fusion method. Using previously trained network structures VGG16, Xception, ResNet50 and Inception-v3 to extract features, a one-dimensional vector of a certain length can be obtained through a global average pooling operation after passing through the intermediate structure of each network, then all one-dimensional vectors are connected into a long vector as the input of the fully connected layer. The output layer outputs the probability that if the grayscale image corresponds to the executable file containing the malicious program.

The model FCNNMD was finally trained on the training set B and the model performance was evaluated on the validation set V.

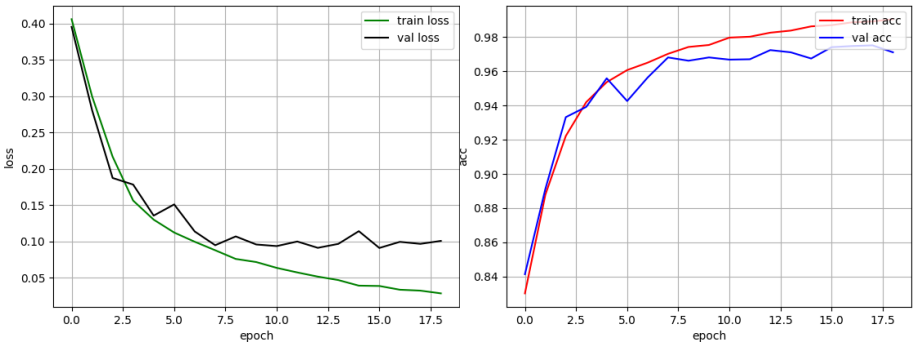


Figure 8. Loss and accuracy curve of custom model (training set B, verification set V)

The left figure of Figure 8 shows that with the increase of the number of epoch, the loss value of the model on the training set B and the verification set V shows a downward trend. From the right figure, we can see that the accuracy of the model on the training set B and the verification set is increasing with the increase of the number of epoch. In the 16th epoch, the model has the lowest loss value and the highest accuracy.

Table 3. The evaluation result of fusion model on verification set V

Metric	**training set A**	**training set B**
accuracy	96.65%	97.70%
false alarm rate	3.21%	3.06%
missing report rate	3.49%	1.54%
time overhead	230s	230s

The comparison with the results of single model shows that the performance of the model can be further improved by model fusion (Table 3), and the generalization ability of the model can be enhanced. The model obtained by model fusion has higher accuracy rate, lower false alarm rate and missing alarm rate.

Table 4. The evaluation result on training set B

Metric	**Simple Fusion**	**FCNNMD**
accuracy	97.70%	97.8%
false alarm rate	3.06%	3.05%
missing report rate	1.54%	1.35%
time overhead	230s	73s

As can be seen from Table 4, the simple fusion model has high complexity and low detection efficiency, and the time overhead of the model is basically equal to the sum of the time overhead of the four single models. FCNNMD complexity is greatly reduced, and the detection efficiency of the model is greatly improved, and the overall improvement is 3.15 times. The reason why the efficiency of model detection is greatly improved is that the number of parameters in the model is greatly reduced and the model is simplified by optimization.

In our method, when adjusting parameters in reverse, only the parameters in full connection layer are adjusted without changing the parameters in each network structure, which weakens the effect of model fusion. Sufficient hardware resources allow all models to be loaded into memory. When adjusting parameters in reverse, the parameters of some layers in each network structure can be fine-tuned according to the results of model fusion.

5. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel fusion method based on convolutional neural network for malware detection. The model combines four classical convolutional neural network structures and its experimental results show that this method can effectively improve the accuracy and robustness of the model. The malware detection method based on convolutional neural network can avoid complex feature code extraction work, reduce labor costs, reduce system overhead, improve detection efficiency and detect new viruses. The training of the generative adversarial network in our model is difficult. In the future, we would like to research a more structured generation adversarial network training model to generate higher quality data.

REFERENCES

[1] Sami A, Yadegari B, Rahimi H, et al. Malware detection based on mining API calls: ACM Symposium on Applied Computing, 2010[C].

[2] M. Christodorescu, S. Jha. Static analysis of executables to detect malicious patterns[C].//In Proceedings of the 12th USENIX Security Symposium, 2003:169–186.

[3] A. Sung, J. Xu, P. Chavez, S. Mukkamala. Static analyzer of vicious executables (save)[C].//In Proceedings of the 2004 Annual Computer Security Applications Conference (ACSAC), 2004:326–334.

[4] R. Lo, K. Levitt, R. Olsson. MCF: a malicious code filter[C].\\Computers and Security 14,1995:541–566.

[5] SANTOS I,BREZO F,UGARTE-PEDRERO X,et al.Opcode sequences as representation of executables for data-mining-based unknown malware detection[J]. Information Sciences,2013,231(2013):62-82.

[6] NAKAZATO J,SONG J,ETO M. A novel malware clustering method using frequency of function call traces in parallel threads[J]. IEICE transactions on information and systems,2011,E94-D(11):2150-2158.

[7] Sundarkumar G G, Ravi V. Malware detection by text and data mining[C]. IEEE International Conference on Computational Intelligence and Computing Research. IEEE, 2013:1-6.

[8] Matthew G.Schultz,Eleazar Eskin,Erez Zadok.Data Mining Methods for Detection of New Malicious Executables [C].IEEE Computer Society,2001:38-49.

[9] Sundarkumar G G, Ravi V, Nwogu I, et.al. Malware detection via API calls, topic et al. Malware Detection Systems Baesed on API Log Data Mining[C]. IEEE, Computer Software and Applications Conference. IEEE Computer Society, 2015:255-260.

[10] Fujino A, Murakami J, Mori T, Discovering similar malware samples using API call topics[C]. Consumer Communications and NETWORKING Conference. IEEE, 2015:140-147.

[11] Hyrum S. Anderson, Phil Roth, et al. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models[C] Neuroscience School Of Advanced Studies(NSAS), 2018, 4.

[12] DAHL G E, STOKES J W, DENG L, et al. Large-scale malware classification using random projections and neural networks[C] 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Vancouver, BC, Canada: IEEE, 2013: 3422-3426.

[13] SAXE J, BERLIN K. Deep neural network based malware detection using two dimensional binary program features[C] 201510th International Conference on Malicious and Unwanted Software (MALWARE). Fajardo, Puerto Rico: IEEE, 2015: 11-20.

[14] KOLOSNJAJI B, ZARRAS A, WEBSTER G, et al. Deep learning for classification of malware system call sequences[C] Australasian Joint Conference on Artificial Intelligence. Hobart, TAS, Australia: Springer International Publishing, 2016: 137-149.

[15] TOBIYAMA S, YAMAGUCHI Y, SHIMADA H, et al. Malware detection with deep neural network using process behavior[C] 201640th Annual IEEE Conference on Computer Software and Applications (COMPSAC). Atlanta, GA, USA: IEEE, 2016, 2: 577-582.

[16] PASCANU R, STOKES J W, SANOSSIAN H, et al. Malware classification with recurrent networks[C] 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Brisbane, QLD, Australia: IEEE, 2015: 1916-1920.

[17] Edward Raff, Jon Barker, Jared Sylvester, Robert Brandon, Bryan Catanzaro, Charles Nicholas, et al. Malware Detection by Eating a Whole EXE[C], Neuroscience School Of Advanced Studies(NSAS). 2017, 10.

[18] Goodfellow, I., Bengio, Y., Courville, A.. Deep learning (Vol. 1). Cambridge：MIT press， 2016.

[19] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, L., Wang, G.and Cai, J., 2015. Recent advances in convolutional neural networks. arXiv preprint arXiv:1512.07108.

[20] LeCun, Y. and Bengio, Y., 1995. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10), 1995.

[21] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, L., Wang, G.and Cai, J., 2015. Recent advances in convolutional neural networks. arXiv preprint arXiv:1512.07108.

[22] Khan R U, Zhang X, Kumar R. Analysis of ResNet and GoogleNet models for malware detection[J]. Journal of Computer Virology and Hacking Techniques, 2019, 15(1): 29-37.

[23] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[24] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet classification with deep convolutional neural networks[C]. Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, 2012 : 1097-1105.

[25] SIMONYAN K, ZISSERMAN A J C. Very Deep Convolutional Networks for Large-Scale Image Recognition[J], 2015, abs/1409.1556.

[26] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with convolutions[J], 2015, : 1-9.

[27] IOFFE S, SZEGEDY C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[C]. ICML, 2015 : .

[28] SZEGEDY C, VANHOUCKE V, IOFFE S, et al. Rethinking the Inception Architecture for Computer Vision[J], 2016, : 2818-2826.

[29] SZEGEDY C, IOFFE S, VANHOUCKE V. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning[C]. AAAI, 2016 : .

[30] HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition[J], 2016, : 770- 778.

[31] WANG M, LIU B, FOROOSH H J I I C O C V W. Factorized Convolutional Neural Networks[J], 2017, : 545-553.

[32] RADFORD A, METZ L, CHINTALA S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks[J]. CoRR, 2015, abs/1511.06434.

[33] NATARAJ L, KARTHIKEYAN S, JACOB G, et al. Malware images: visualization and automatic classification[C]. Proceedings of the 8th International Symposium on Visualization for Cyber Security, 2011 : 1-7.

[34] Rathore H, Agarwal S, Sahay S K, et al. Malware Detection Using Machine Learning and Deep Learning[C]//International Conference on Big Data Analytics. Springer, Cham, 2018: 402-411.

[35] Sewak M, Sahay S K, Rathore H. Comparison of deep learning and the classical machine learning algorithm for the malware detection[C]//2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). IEEE, 2018: 293-296.

[36] Gibert D, Mateu C, Planes J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges[J]. Journal of Network and Computer Applications, 2020: 102526.

AUTHORS

I am a postgraduate student of the Institute of Information Engineering, University of Chinese Academy of Sciences, Majoring in deep learning and security.