DOI: 10.53469/wjimt.2026.09(02).04

Leveraging Convolutional Neural Networks for Malware Detection: A Review of Techniques, Architectures, and Emerging Trends

Hai Mei, Li Zhou

Jincheng College, School of Computer and Software, Sichuan University, Chengdu 611731, Sichuan, China

Abstract: ***With the modernization of the Internet and society, people often use some mobile phone applications to facilitate their lives. But since most of these apps require sensitive information from users, malware has been developed by malicious people to infect users' phones, especially on the Android platform. In order to detect whether a file after download is malware, this paper takes a static detection method. By decompiling and uncompressing the downloaded APK file, extracting the software rights from it, converting the rights into feature graphs, and putting them in a neural network to test. After four fold cross-testing, the average accuracy on the validation set was 88.205 percent. The group that trained the network with the highest accuracy of its data sets was then selected as the training set for the final network, and then tested on a test set, which showed that this method worked better than traditional machine learning methods.***

Keywords: Convolutional Neural Network; Android Malware; Cross-validation; Permissions.

1. INTRODUCTION

Due to the modernization of society and the popular application of the Internet, people are increasingly dependent on the use of mobile phones. People often use some of the mobile phone applications to facilitate their own lives, but because most of these applications require the user's mobile phone number, social networking accounts, ID card number and other sensitive information, so some malicious people develop malicious software to infect the user's phone. At the same time, because of the open nature of Android, many customers will install malware without knowing it. Once we install malware on our phone, these people can steal data, which will have a huge hidden danger to our information security.

In order to detect whether the downloaded software is malicious software directly after download, this paper adopts a static detection method by decompiling and uncompressing the downloaded APK file. Software permissions were extracted from AndroidManifest.xml. Each permitted feature was treated as a feature, converted through a heat-coded encoder and then turned into feature pictures and categorized in a convolutional neural network. Experiments have shown that this method has a higher recognition accuracy than traditional machine learning.

2. OVERVIEW OF RELEVANT THEORIES

2.1 Single Hot Coding Overview

The unique hot coding is also called one-hot coding. The specific use of the unique hot coding method is to quantize the non-digital features once, and convert the non-digital features into 01, 10 binary codes. This extends the value of discrete features directly to European space, where the value of the discrete features corresponds to the points of European space, and makes the calculation of the distance between the features more reasonable, thereby enhancing the usability of the model.

2.2 Overview of Convolutional Neural Networks

Convolutional neural networks are pre-feedback neural networks composed of input layers, output layers, convolution layers, activation functions, pooling layers, and full connectivity layers. The structure of the classic Neural Network LeNet-5 is shown in Figure 1:

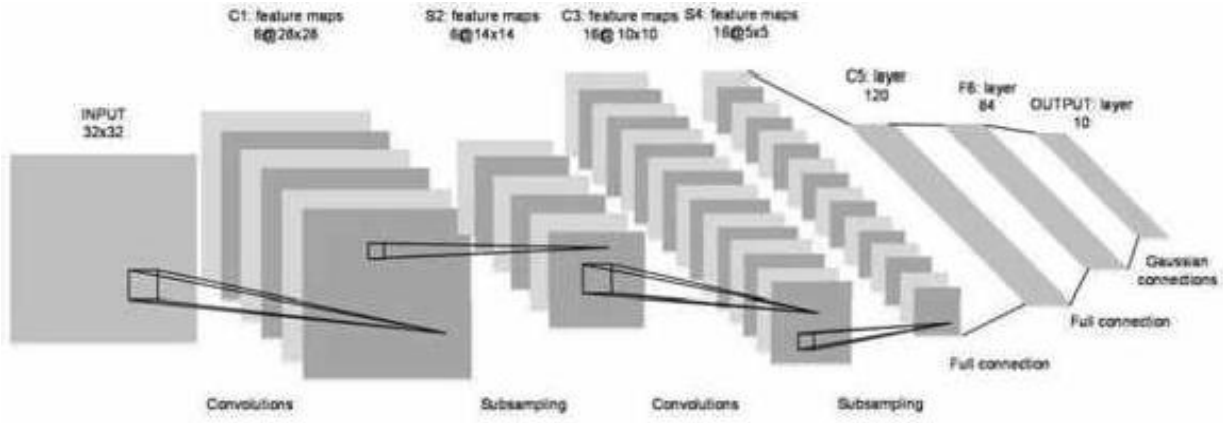

Figure 1: LeNet-5 Composition

Among them, the convolutional layer can extract the feature information of the feature diagram and generate the feature matrix by convolutional computation. For example, from C1 of the above diagram, the image of 32×32×1 imported from the input layer is first convolutional and processed into a feature matrix of 28×28×6 imported into the next layer. Pooling layer Pooling operations can reduce the feature matrix and effectively solve the situation of excessive computation. For example, S2 of the above diagram, the 28×28×6 feature diagram transmitted by the above layer is pooled and processed into a 14×14×6 feature picture. After the convolution-pooling operation, the multidimensional feature matrices are transformed into one-dimensional characteristic vectors, which are introduced into the full connection layer.

3. PROCESSING OF DATA SETS

3.1 Data sources

In the selection of data, choose to use the public data set of MalDroid-2020 on CICDataset, There were more than 17,341 Android examples, chosen 1,150 positive samples and 1,175 negative samples, extracted the AndroidManifest.xml file from the APK through decompilation and decompression, and extracted the permission features in the APK from that file as an initial dataset.。

Then combine the permissions features of each APK to get a total of 200 different permissions features, and use a binary vector to transform each feature.The eigenvector of 1×400 is obtained, and then this eigenvector is transformed into a 20×20 feature map. The value of a pixel in the feature map is the eigenvalue of the software corresponding to one permission [3]. Finally, after normalizing their feature graph, a training, test and validation set is obtained. A feature diagram of the positive and negative samples, as shown in Figure 2:

Figure 2: Characteristics of positive samples (right) and negative samples (left)

3.2 Splitting the dataset

For the subsequent folding test verification, a total sample of 2325 data were divided into 4 samples on average, including 2094 positive samples and 265 negative samples, which were counted as A, B, C and D datasets. Finally, the remaining 231 samples were used as the test set, of which 115 positive samples and 116 negative samples were recorded as the E dataset.

4. MODEL TRAINING

4.1 Building a Convolutional Neural Network

Since Android software permission calls need to be related to each other, picture size issues need to be considered. After many experiments, in convolutional neural networks, only one convolutional - pooled layer is used, and two full-connected layers are reasonable. The resulting network is shown in Figure 3:

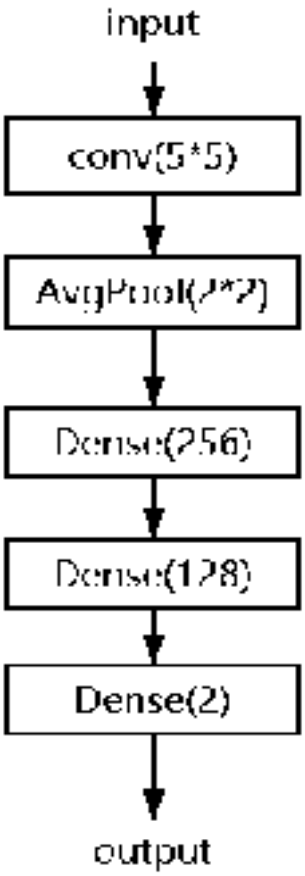

Figure 3: Network structure

First, consider the size of the picture to be entered. After many experiments, it is reasonable to adjust the picture to the size of 32×32. At the same time, to ensure correlation between permission calls, the input image here undergoes a 5×5 convolutional operation and increases its dimensionality, increasing the original number of channels to 6 channels. Since the original image entered was a two-dimensional image, considering that some features might be lost using the maximum pooling operation, after many more experiments, the second layer used an average pooling operation of 2×2 to reduce the size of the feature image obtained after the convolution operation to half the original size. In the remaining fully connected layers, the number of 256 and 128 neurons, which produce faster runs and higher output results, were chosen after experimentation. After many experiments on the other parameters, a reasonable parameter selection was obtained, as shown in Table 1.

Table 1: Other parameters in convolutional neural networks

Learning rate	0.07	Momentum	0.9
Weight decrease factor	0.005	Classifier	Softmax
Optimization method [6]	SGD	Loss function	Cross Entropy Loss
Activation Function	Hot stove		

4.2 Training outcomes

Three of these were selected sequentially as training sets and one as validation sets during training. The result is shown in Table 2:

Table 2: Results of quadruple cross-examination

Training Set and Test Set	Training set loss value	Training set results (accuracy)	Test set results (accuracy)
A, B, C for the training set, D is the validation set	0.2166	91.79%	88.89%
A, B, D for the training set, C is the validation set	0.2012	92.80%	90.84%
A, C, D for the training set, B is the validation set	0.2397	90.13%	90.46%
B, C, D for the training set, A is the validation set	0.2536	90.76%	82.63%

The set of data sets with the highest accuracy was selected for training and testing. The resulting loss image of the training and test sets is shown in Figure 4, which shows that approximately at round 40, the neural network begins to smoothly decline and converge to a minimum.

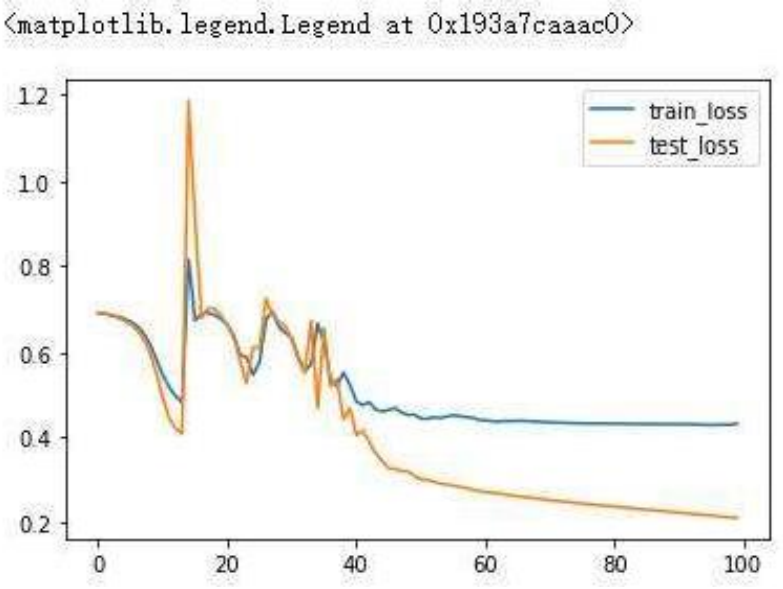

Figure 4: Loss image of training set and validation set

5. EXPERIMENTAL RESULTS ANALYSIS

5.1 Evaluation indicators of the model

Traditionally, true positive rate (TPR), false positive rate (FPR), accuracy, precision and F-Score are used to evaluate a model. Here, remember TP as the amount of malware that was predicted to be malware, TN as the quantity of benign software that was predictedly predicted to have been benign, FN as the numbers of malware which was predicted as benign and FP as the percentage of benignic software that was anticipated to have been malware [1]. The formula for these indicators is:

$$\text{TRP} = \frac{\text{TP}}{\text{TO+FN}} \tag{1}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP+TN}} \tag{2}$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FN+FP}} \tag{3}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{4}$$

$$\text{F} - \text{Score} = \frac{2\times\text{Precision}\times\text{TPR}}{\text{Precision+TPR}} \tag{5}$$

5.2 Experimental results and analysis of different algorithms

Using traditional machine learning algorithms to compare, the results are shown in Table 3:

Table 3: Test results of different algorithms

Using Models	TPR	FPR	Accuracy	Precision	F-Score
CNN	0.7328	0.0435	0.8442	0.9444	0.8252
MLP	0.7758	0.1217	0.8268	0.8653	0.8182
Logistic regression	0.7241	0.0957	0.8139	0.8842	0.7962

As can be seen from Table 3, on the same test set, the use of convolutional Divine networks can be used in guaranteeing TPR, While the scores for Accuracy, Precision, and F-Score were higher, FPR was also lowest, so using convolutional neural networks on this dataset was significantly better than traditional machine learning classification algorithms.

6. CONCLUSION

With the growing market share of Android and the increasing use of the Internet, there is also an increasing number of malware developed to illegally obtain users' information and interests by means of, for example, decompilation and uncompressing downloaded APK files. Extract the privilege characteristic from AndroidManifest.xml, and preprocess the privilege characteristic by using the single hot code and changing the characteristic vector into the characteristic graph. On this basis, a convolutional neural network application tool for detecting malware was implemented. This paper uses a static analysis method,

REFERENCES

[1] Jingping Li. A method for Android malware detection based on privilege [J]. Journal of Northwest University for Nationalities (Natural Science Edition), 2017, 38 (02): 9-13.
[2] Xiaochun Hu, Chengyu Zhu, Yan Chen. Study and analysis of deep convolutional Divine network model [J]. Information Technology and Informatization, 2021 (4): 107-110.
[3] Jie Liang, Jiahao Chen, Xueqin Zhang, Yue Zhou, Jiajun Lin. Anomaly detection based on unique thermal coding and convolutional Divine network [J]. Proceedings of Tsinghua University (Natural Sciences Edition), 2019, 59 (07): 523-529.
[4] YishengFu, Tianliang Lu, Zeliang Ma. One-Hot based CNN malicious code detection technology [J]. Computer Applications and Software, 2020, 37 (01): 304-308,333.
[5] Yuwei Zhang, Yingchun Huang. Research on Malware Classification and Identification Based on Machine Learning [J]. Science and Technology Information, 2018,16 (30): 9-11.
[6] Xuetao Zhang, Meng Sun, Jinshuang Wang. A fast multi-granularity detection approach for Android malicious code based on operation code [J]. Journal of network and information security, 2019, 5 (06): 85-94.
[7] YueTeng, Tianbao Wang. Research on Android Malware Detection Model [J]. Science and Technology, 2017 (02): 51, 61.