MFENet

MFENet: An 1D CNN-based Model for Automatic Feature Extraction and Fusion

黎洋 朱会娟
江苏大学计算机科学与通信工程学院

优胜奖

Abstract

As Android phones become the mainstream of smartphones, more and more malicious applications are being developed to attack the system and steal user information. Many researches based on machine learning can automatically detect Android malware currently. In spite of that, these methods require manual feature selection, which relies on experience and may loss major features. In this paper, we propose a tiny Android malware detection model named MFENet. This model is based on the one-dimensional convolutional neural network (1D CNN). The core of the model is MFEBlock, which can extract key features and fuse them automatically. This model got 94.62% accuracy on the test dataset, which achieves the outstanding results for Android malware detection on an open dataset.

NETWORK STRATEGY

1. Pretreatment method

As Android software has many features, if they are all placed in the same channel, the training cost of the entire network will be seriously increased. If they are evenly placed on multiple channels, the training speed of the network will be greatly accelerated, and no information will be lost.

Based on this idea, we propose a preprocessing method. Assuming that the number of features is m, then square m and round up to get p as the number of channels to be divided. However, n ($n = p * p$) is less than or equal to m, so we need to fill it with 0. In this process, the original feature vector (size m) can be multiplied by an identity matrix of size $m * n$ to obtain a new feature vector of size n. To prevent the information of the last channel1 from being filled with 0, we insert 0 into the beginning and end of the original feature vector. This can be achieved by setting the index of the diagonal element. The specific algorithm is as follows:

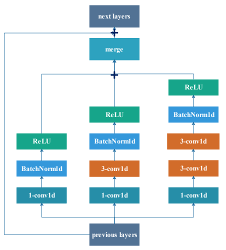
$$k = \frac{1}{2}\lfloor n - m \rfloor$$

Which k is the index of diagonal element. The following is an example of this algorithm.

$$[1\ 0\ 2\ 3\ \ldots\ldots 0\ 4\ 3\ 1] * \begin{bmatrix} 0\ 1\ 0\ \ldots\ldots 0\ 0\ 0 \\ 0\ 0\ 1 \ldots\ldots\ 0\ 0\ 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ 0\ 0\ 0\ \ldots\ldots 0\ 1\ 0 \end{bmatrix} = [0\ 1\ 2\ 3\ \ldots\ldots 0\ 4\ 3\ 1\ 0]$$

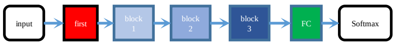
2. MFEBlock

The key to our model is the block. For the purpose to extract more details form the raw features, we design a novel block called MFEBlock (Multi-scale Features Extract and Fusion Block). This block is composed of three paths, one merged layer and one residual pathway. The first path has the minimum receptive field and its training speed is the fastest of the three. The second path has a good receptive field and faster training speed. The last path has the maximum receptive field, so it can extract most information but its training speed is lowest among three. The merged layer consists of a lx1 kernel lD convolution layer, a batch normalization layer and a ReLU activation layer, which is similar to the first way. The previous layer x went through 3 different routes and got y_1,y_2, y_3. Then we add them up to get y, which would be the input of the merged layer. If the block does not work, the previous layer x can also go through the residual pathway. Therefore, this block can be stacked in multiples while maintaining a minimal amount of parameters and high training speed.



3. MFENet

We named this Network as MFENet (Multi-scale Features Extract and Fusion Network), as shown Fig. 3 and Table I. In this network, feature extraction and classification are split into two principal parts. In feature extraction, we first use a 3x1 kernel 1d convolution layer to extract low-level information, a batch normalization layer and a ReLU activation layer. Then we use three MFEBlocks to obtain different scale information. We use global pooling technology to convert the extracted feature maps into feature vectors. Finally an FC layer is used for the final classification.



4. Trainging Methodology

Our networks were trained using the deep learning framework Pytorch and graphics card is GTX1050. We use Adam as the network's optimizer and cross-entropy as the network's criterion. During the training process (as shown in Algorithm 1), we record the time spent in each epoch. The final train loss, test loss, train accuracy and test accuracy will also be taken down as the experimental results.

```
Algorithm 1 MFENet training process
Input:
 Ds : the raw dataset
 Ep, Bs : training epochs and batch size for MFENet
Output: A trained MFENet model
Function:
1.  For i =1 to Ep do
2.      For b =1 to size(Ds)/Bs do
3.          get the data x and label y from dataset
4.          get predictions z from MFENet
5.          $L = -[y\log z + (1-y)\log(1-z)]$
6.          use back propagation algorithm to update W
and b
7.      end For
8.  end For
```

Experiment

1.Dataset

Our experiments are based on a public data flow dataset supported by Avdiienko, which includes 3733 data flow features extracted from 17897 samples (including 2799 negative samples and 15098 positive ones).

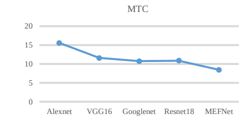
2. Experiment Result

We evaluate our network MFENet with other models from three indicators: MTC, loss and accuracy.

MTC (Mean Training Cost) is a new indicator defined by us to judge the comprehensive performance of training time and parameter memory indicators. It can expressed as :$MTC = \sqrt{M * T}$, where M is the memory occupied by the models and theirs parameters, T is the time spent training this model every epoch.

Taking accuracy as an indicator is currently a popular choice, we also compare our model MFENet with other networks. In the testing accuracy, MFENet got 94.62% score and is the highest score in these models. To make the score more credible, all these scores are collected in the final epoch and are the average in 5 folds cross-validations.

In summary, the experiment shows that MFENet performs best among the models we list. Not only the test accuracy and loss, but also the performance of MTC is the best.



Model	TrainLoss	TestLoss	TrainAcc	TestAcc	MTC
Alexnet	0.0226	0.3514	99.10	94.25	15.55
VGG16	0.0352	0.3123	98.70	93.89	11.59
Googlenet	**0.0224**	0.3046	**99.16**	94.18	10.73
Resnet18	0.0277	0.3440	98.89	93.31	10.86
MFENet(ours)	0.0227	**0.2697**	99.11	**94.62**	**8.45**

3. Horizontal Comparisons

To better demonstrate the effectiveness of our model, in this section, we have trained a variety of classic machine learning models to compare MFENet's detection capabilities with those of these models on Android malware. As can be shown in following table, MFENet got the supreme performance.

Model	**Accuracy**
Gaussian Naive Bayes	87.11%
LogisticRegression	92.59%
K-NearestNeighbor	90.3%
Decision Tree	92.89%
Adaboost	92.49%
Gradient Boost Decision Tree	93.16%
MFENet(ours)	**94.62%**

Conclusion

In this paper, we use 1D convolution, so the malicious features do not need to convert to the matrix and reduce the risk of information loss. To retain more information, we abandon the pooling layers. Moreover, we design a novel block, which can extract multilevel features and fuse them effectively. In order to better evaluate the performance of the model, we have defined a new indicator MTC. Regardless of test accuracy, test error, or MTC, our proposed model achieves the outstanding results on this public data set.

Main References

[1]J. Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning," Springer, 2018.
[2]Q. H. T. R. Laboratory. "Quick Heal Annual Threat Report 2020,"https://www.quickheal.co.in/resources/threat-reports,2020
[3]K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition." pp. 770-778.
[4]H. Cai, N. Meng, B. G. Ryder, and D. Yao, "DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling," IEEE Transactions on Information Forensics and Security, vol. 14, no. 6, pp. 1455 - 1470, 2019.
[5]V. Avdiienko, K. Kuznetsov, A. Gorla, A. Zeller, and E. Bodden, "Mining Apps for Abnormal Usage of Sensitive Data." Proc. 37th IEEE International Conference on Software Engineering, pp. 426-436, 2015.

Contact us

Author：Li Yang
Phone：17305292396
E-mail：lyml6@foxmail.com