Effective feature selection for Deep learning based android malware detection

Seongyun Jeong
UNIST
jsy01@unist.ac.kr

Abstract

As the Android system is widely used, many Android Malware are also being created even at this moment. To solve this problem, many deep learning-based studies have been presented to detect Android malware. However, in most studies, They select inefficient feature selection without considering the dependency relation of API calls and permissions, which are data used for training model. So, in this paper, we propose a new efficient feature selection scheme, removing redundant permissions. We conduct four deep learning accuracy check experiments by separating the features, including a new scheme. And we achieve the 99.2% classification accuracy using DNN. This paper first suggests the importance of considering the dependency relation in the dataset.

1 Introduction

Android operating system is the most widely used operating system. According to the statcounter[3], the Android operating system occupies 70.96% of the total mobile operating system market. However, as the market has grown, so has the proportion of malware, a generic term for malicious software. According to Nokia's Threat Intelligence Report[5], Android malware samples continue to grow over and over and in 2020, There are now close to 27 million Android malware samples, which represents a 17.4% year-over-year increase.

The most accurate malware detection method is to be handled manually by a person. But this method is too expensive; Since more than 10,000 new malware are released every day[8], it should be possible to detect only the app's APK file. Because of this situation, numerous studies are being conducted on deep learning based Android malware detection through data feature extraction from APK file. Whether the method is static or dynamic, The most important point is that the extracted data should contain a lot of information about the malicious behavior of the app. Some part of it should represent information about malicious behavior.

In the data extraction of Android malware detection to satisfy the rich information[10], the most used analysis method is android characteristic-based method; collect data such as permission, API calls, intents by decompiling and find some symbols and function calls. These data are considered as feature vectors used in the machine learning models. Therefore, the two most important things in detection are well-extracted data and a model that can learn it well. Many studies[11] focus on learning models rather than data extraction and processing. In most cases, the same tool[7] is used in the data extraction process, and data processing is performed by bundling information such as permission, API call, and intent into the same feature vector.

The problem is that each feature is not all independent; Some of the API calls and permissions have complex dependencies. A specific API call requires user's private data, and permission is required to execute the API call. In addition, the permission desired by the API call is not set to one, and differs depending on the Android version. If we ignore these dependency relations and conduct machine learning, it will be difficult to obtain high accuracy.

So, for high accuracy of deep learning. We propose an efficient feature selection scheme; Reducing permission features which are already included in the API call features. We suppose that the API call had more information about the app's behavior than permission. In other words, the priority of the API is higher than permission, and the permission that can hinder learning is removed.

We use dataset[12] consisting of feature vectors of 215 attributes extracted from 15,036 application(5,560 malware apps from Drebin[6] project and 9,476 benign apps). With this dataset, we conducted four experiments: (1) features with permis-

sion and API call at the same time (original version), (2) features with only permission, (3) features without permission, and (4) reduced version of features. In our design, It achieved up to 99.2% accuracy with Deep Neural Network.

The main contribution of this work is as follows:

- We point out the problems of feature selection in existing studies. We suggest that when API call and permission are used simultaneously as a dataset, learning should be conducted in consideration of the dependency of the features.
- We propose the ways to overcome learning rate errors due to dependency. Through this method, We reach 99.2% accuracy from the existing 98.6% accuracy.

You can obtain the training dataset files and our deep learning model for testing dataset from https://github.com/Seongyun-Jeong/Effective-FS-for-Android-Malware.

2 Background

2.1 API (Application Programming Interface)

The Android platform provides Application Programming Interfaces(API) that allow applications to communicate smoothly with the Android system. Thanks to this, when developing an android application, we can access users' private data access the Internet, and so on without an understanding of the Android system. So, API calls reflect the functionality and behavior of an application.

2.2 Permission

Applications should not access freely to the user's private data. The Android system can keep the system safe only when the application restricted to access the user's private information or Internet and so on. So, Android use permission system for this. Android apps require some permissions to perform sensitive behavior. Permission information should be declared in the *AndroidManifest* file in advance. For example, If an application wants to perform network operations, application's manifest must include *INTERNET* and *ACCESS_NETWORK_STATE* [2].

3 Current problems in feature selection

Let's assume that application wants to call *getLine1Number* API call. This API Returns the phone number string. In general application, it rarely uses a phone number. If there is an application that uses this API call, we can classify it as a high-potential malware group.

Since it access sensitive information, the Android system will require permissions. Refer to official API reference[4], This API call requires *READ_SMS* or *READ_PHONE_STATE* or *READ_PHONE_NUMBERS*. It looks weird to declare one of the various permissions to execute one API call. There are 2 reasons; One is as the Android version was updated, required permission was changed, And other reason is that one permission is the higher concept of another permission.

When the Android version is below 11 (Target SDK API < 30), System require the *READ_PHONE_STATE*. But, over the Android version 11(Target SDK API > 29), It changed to require the *READ_PHONE_NUMBERS*. If the application made before the android version 11, it will use former's permission. And if it made after the version 11, it will use latter or both permission because of compatibility. And as the name suggests, *READ_SMS* permission includes more functions than the other two permissions. It is not just access to phone numbers, but also access to SMS.

The important point is that these permissions are defined for an API call such as *getline1number*. If we put them in the same feature vector and proceed deep learning, the model will keep learning about strange permission relations like *READ_SMS* and *READ_PHONE_NUMBER* even if the most important feature is *getline1number* in this scope. Table 1 summarizes the number of possible cases in this situation. If high-potential malware is determined only by API call, redundant permission will interrupt learning and reduce the accuracy of deep learning.

4 The proposed Method

Our proposed solution is shrunk permission features which has same dependency relation on API. To do that, we have to investigate the relation between permission and API. We manually check the dependency by referring to the official API documentation[1]. Unlike Section 4, There are many API call features that was used in class units; contain all member functions. So, we collect total 12 permissions that has same dependency relation on API. Table 2 summarizes the 12 permissions we selected and APIs and intents that related to

Table 1: Possible cases in example situation; Determining potential malware involves with an API call

READ_PHONE_STATE	READ_PHONE_NUMBERS	READ_SMS	getLine1Number API call	High-potential Malware?
0	0	1	1	1
0	1	0	1	1
0	0	1	1	1
...				
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0

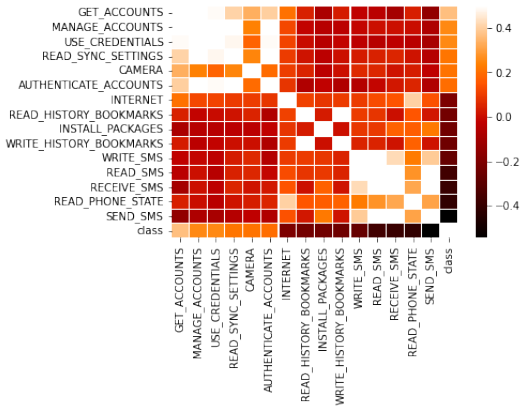

Figure 1: Heat-map image of permission.

them.

Figure 1 is top correlation coefficient heat-map image of permission. As shown in Figure 1, among the collected 12 permissions, 10 permissions have correlation coefficient greater than 0.2 in absolute value compared to True table. This proves that the permission we shrink plays a large part in the existing learning. If learning accuracy is better even though after remove these permissions, we can prove our thoughts are correct.

5 Experiments and Analysis

We conduct total 4 deep learning accuracy test:

1. features with permission and API call at the same time (original version)
2. features with only permission
3. features without permission
4. reduced version of features

In our hypothesis, The accuracy of (3) is higher than (2), because APIs represent more detailed app's behavior than permissions. And the accuracy of (1) is higher than (3), because even though permission represent less about app's behavior, There are some sensitive parts that API cannot represent. Finally, the accuracy of (4) is higher than (1) because we reduce the redundant permission features in (4). To sum up, the accuracy would be in the order of (4)-(1)-(3)-(2).

5.1 Experiment Environment

We conduct deep learning training and test in Google Colab. All data processing process is in my GitHub. We separate training and validation dataset by 8:2. Our deep learning model is DNN (Deep Neural Network): 2 hidden layers and activation function of hidden layer is ReLU. At the end of the layer, we use sigmoid activation function to binary classification. And, We use loss function as binary_crossentropy and use adam optimizer. The reason why we choose DNN is that it is easy to implement and have higher accuracy than other models that use random-forest algorithm or LSTM (Long Short Term Memory). Also, there are many previous studies[11] that use various deep learning models, we mainly focus on data selection and apply to simple deep learning model.

5.2 Features with permission and API call

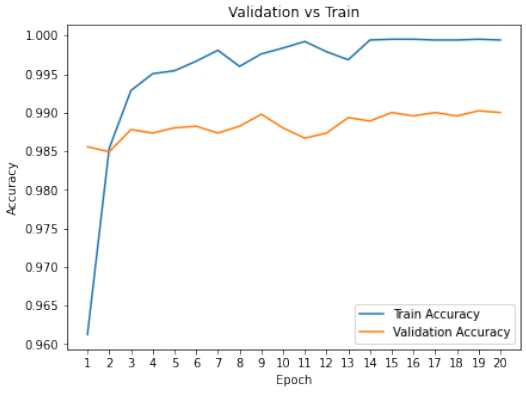

Figure 2: Permission + API : Validation VS Train Accuracy graph

Since the original dataset also performed very well in the data processing and feature selection process, the accuracy is very high: As shown in

Table 2: Removed Permission in dataset. Total 12 number of permission deleted.

No.	Removed Permission	Related API calls, intents
1	SEND_SMS	android.telephony
2	READ_PHONE_STATE	android.telephony
3	GET_ACCOUNTS	ACCOUNT_MANAGER
4	RECEIVE_SMS	android.telephony
5	READ_SMS	android.telephony
6	MANAGE_ACCOUNT	ACCOUNT_MANAGER
7	WRITE_SMS	android.telephony
8	AUTHENTICATE_ACCOUNT	ACCOUNT_MANAGER
9	INSTALL_PACKAGES	android.content.pm.Package
10	INTERNET	NETWORK
11	RECEIVE_BOOT_COMPLETE	android.intent.action.BOOT_COMPLETED
12	DELETE_PACKAGES	android.content.pm.Package

Figure 2, Accuracy fluctuates between 0.985 and 0.99. Average accuracy is 98.8%. To examine the trend of the accuracy of the validation dataset, training was repeated 20 times.

5.3 Features with permission

As shown in Figure 3, Accuracy fluctuates between 0.94 and 0.96. Average accuracy is 95.4%, which is lower than Permission + API one. But, It is interesting that the accuracy is higher than we thought even though conducting deep learning training with only permission features. Since we thought that permission is an abstract feature compared to API that directly represents the action, I expected that the accuracy would be low.

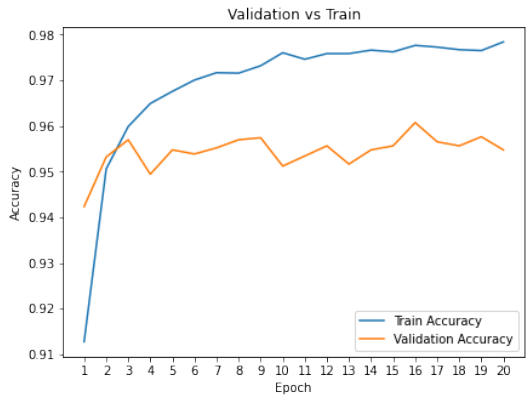

Figure 3: Permission : Validation VS Train Accuracy graph

5.4 Features with API call

As shown in Figure 4, Accuracy fluctuates between 0.97 and 0.98. Average accuracy is 98.0%, which is lower than Permission + API one, higher than Permission only. Through this result, we can say that API represents the app's behavior better than permission, and at the same time, permission is also indispensable feature in malware classification. In Section 3, we said that some permission features can reduce the accuracy of deep learning, but we can check that others are representative of the app's behavior.

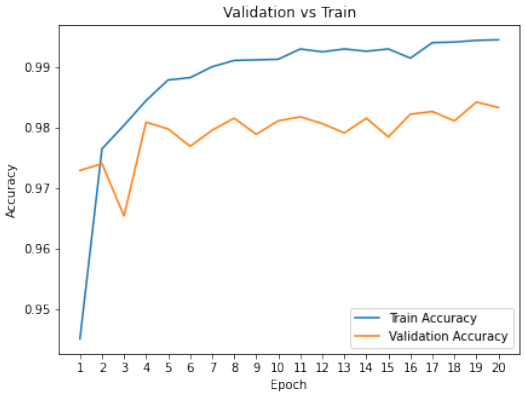

Figure 4: API : Validation VS Train Accuracy graph

5.5 Reduced version of features

As shown in Figure 5, Accuracy fluctuates between 0.99 and 0.994. Average accuracy is 99.2%, which is the highest accuracy. This result prove our hypotheses. Even though the permissions we removed have high correlation coefficients, we found that removing them in the entire feature vector helps in model training. Deep Learning through various features is a very important part in malware detection, but it is also important to consider the dependency relation of each feature.

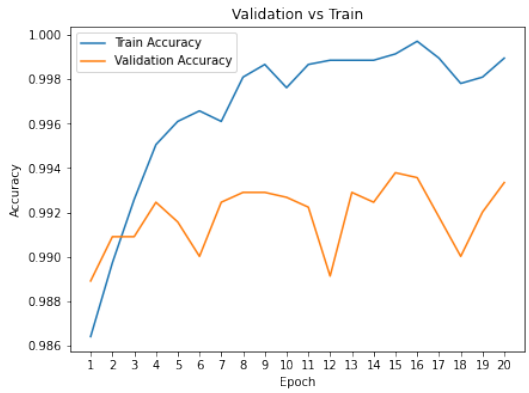

Figure 5: Reduced Permission : Validation VS Train Accuracy graph

6 Conclusion

6.1 Limitation

The accuracy may result differently depending on the dataset and model. Recently, There are many android malware detection papers using various learning models. Depending on the specific model, there may be a model that automatically solves the problems that proposed in this study. For example, a model that uses a well-processing API call dataset without using permission[9] can show high accuracy without dependency issues. Manually collecting dependency relations is also problematic. Despite the official Android documentation provides information on API, it is hard to look at a large amount of API documentation and understand the correlation with permission. We need to create a program that automatically analyzes the required permission by looking at the API defined function.

6.2 Result

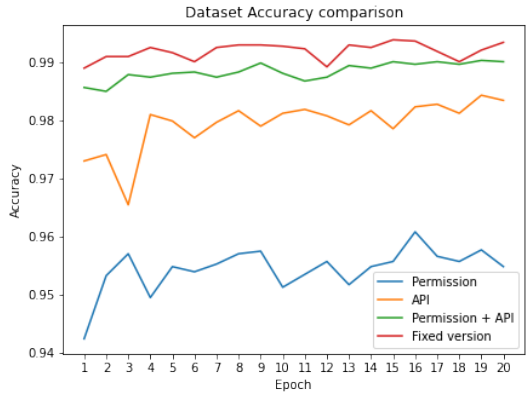

Figure 6: All dataset of Validation Accuracy graph

Following Figure 6 is compilation of the validation accuracy of previous datasets. All graphs are divided without overlapping parts. In Android, permission and API are good features to classify malware. Among them, in particular, API can represent more about the app's behavior than the permission. Although many previous studies put permission and API call into the same feature vector and proceed with learning, this study proposes that learning accuracy can be improved by selecting features in consideration of the dependency relation of API and permission.

References

[1] Android api reference nbsp;: nbsp; android developers. *Android Developers*.

[2] Connect to the network nbsp;: nbsp; android developers. *Android Developers*.

[3] Mobile operating system market share worldwide. *StatCounter Global Stats*.

[4] Telephonymanager nbsp;: nbsp; android developers. *Android Developers*.

[5] Threat intelligence report 2020. *Nokia*.

[6] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. 2014. Drebin: Effective and explainable detection of android malware in your pocket. 14:23–26.

[7] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. 2014. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm Sigplan Notices*, 49(6):259–269.

[8] G Data. 2022. Mobile malware report - g data. *O melhor G DATA de todos os tempos - Versão de teste 2020*.

[9] Zhuo Ma, Haoran Ge, Yang Liu, Meng Zhao, and Jianfeng Ma. 2019. A combination method for android malware detection based on control flow graphs and machine learning algorithms. *IEEE access*, 7:21235–21245.

[10] Ya Pan, Xiuting Ge, Chunrong Fang, and Yong Fan. 2020. A systematic literature review of android malware detection using static analysis. *IEEE Access*, 8:116363–116379.

[11] Junyang Qiu, Jun Zhang, Wei Luo, Lei Pan, Surya Nepal, and Yang Xiang. 2020. A survey of android malware detection with deep neural models. *ACM Computing Surveys (CSUR)*, 53(6):1–36.

[12] Suleiman Y. Yerima and Sakir Sezer. 2019. Droidfusion: A novel multilevel classifier fusion approach for android malware detection. *IEEE Transactions on Cybernetics*, 49(2):453–466.