Collective intelligence: Decentralized learning for Android malware detection in IoT with blockchain

서울과학기술대학교 컴퓨터공학과 진호천

Depart. Computer Science, Seoul National University of Science and Technology

CONTENTS

1. Introduction
2. Related work
3. Preliminaries
4. System model
5. Performance Evaluation
6. Conclusion

Abstract

- ✓ The broad significance of Android IoT devices lies in their flexibility and hardware support capabilities through the introduction of diverse applications in almost all areas of daily life. This omnipotent applicability provides an incentive for more malware attacks.

- ✓ Authors propose a novel neural network (LNN) for local training. (ii) A new smart contract is proposed to realize the aggregation process on the blockchain platform. The LNN model analyzes various static and dynamic characteristics of malware and benign software, while smart contracts use aggregate characteristics stored in the local model to verify malicious applications in the upload and download process in the network.

- ✓ The proposed model not only uses the decentralized model network to improve the accuracy of malware detection, but also uses the blockchain to improve the effectiveness of the model.

Introduction

- ✓ A range of advanced electronic devices are controlled by the powerful Android platform, enabling smart devices such as sensors, smart phones, smart watches, smart washing machines and so on. These electronic devices, encourage people to store and share their personal and confidential information.
- ✓ Due to the common Android platform, these devices are an intensive target for malicious applications to harm users.
- ✓ Attackers use the Android system to directly affect the privacy and security of users. Malicious applications can adversely affect not only the intended node, but even other connected devices that use the shared network.
- ✓ Therefore, there is an urgent need for an evolutionary approach and framework to detect malware applications in a timely manner.

Figure 1. An illustration for the integration of IoT smart devices connected on a common network via Android application platform.

Introduction

- ✓ 1) This paper designs a smart contract to provide a secure downloading and uploading mechanism for Android applications.
- ✓ 2) This paper proposes a framework that integrates federated learning and blockchain for better malware detection of Android applications and information sharing across networks.
- ✓ 3) Proposes an enhancement of a multi-layer deep learning model that can extract multiple types of malware characteristics and distribute training tasks across a blockchain network for better prediction.
- ✓ 4) By providing multi-level deep learning and secure data sharing via blockchain, we have conducted extensive empirical analysis to demonstrate the importance of the proposed approach.

Related Work: Android Devices Security For IoT

- ✓ Because of the flexibility and accessibility of the Android operating system, it dominates the mobile market on billions of devices worldwide. At any given moment, millions of Android apps are available for end-users to install through various app stores such as Google Play.
- ✓ With the popularity of IoT devices in the digital world and the widespread use of iot platforms, users' personal information is stored in recent years, the Internet of Things network has received a lot of attention from attackers. Many researchers have highlighted the importance of malware detection and IoT security for Android devices, with machine learning as a promising solution.
- ✓ Along with machine learning algorithms, researchers are also beginning to use innovative blockchain technology to protect the underlying IoT smart devices. Blockchain deployment ensures transparency, decentralization, verifiability, fault tolerance, auditability, and trust.

Related Work: IoT Platform

- ✓ The IoT platform is used to build applications that monitor iot devices. These platforms provide developers with the ability to quickly build, test, deploy, and iterate on IoT-Specific applications. The global market share of IoT platforms will reach $74.74 billion by 2023, according to the G2 Rating Database, a technology report. The reason behind this growth is the huge demand for IoT devices and other components. As a result, leading technology stakeholders strive to win the race to provide sustainable IoT platforms.

- ✓ These platforms act as a central hub where other smart devices interact and the cloud is used to synchronize device state. These devices collect physical information and send events to the cloud to trigger other events.

- ✓ Divide this section into three parts
- ✓ i) Static analysis, which consists of two methods, the first method is based on permissions, the second method is API calls.
- ✓ ii) Dynamic analysis for real-time feature extraction of mobile phones.
- ✓ iii) Mixed analysis combining static and dynamic features.

Static Analysis

- ✓ Static analysis allows you to examine the behavior of your application without executing it. Content-based analysis can distinguish between benign and malignant applications.
- ✓ Permission-based analysis ensures that users' sensitive information is restricted to only real users. In fact, permissions are the most effective static feature because attackers need to apply for permissions in order to achieve their malicious goals. Previously, when the App was installed, it would ask the user for some requested permissions. After authorization, the application installs itself on the device.
- ✓ API calls :API stands for application interface. Applications use API calls to interact with the Android framework. Some work targets API calls as a promised feature to investigate malicious behavior.

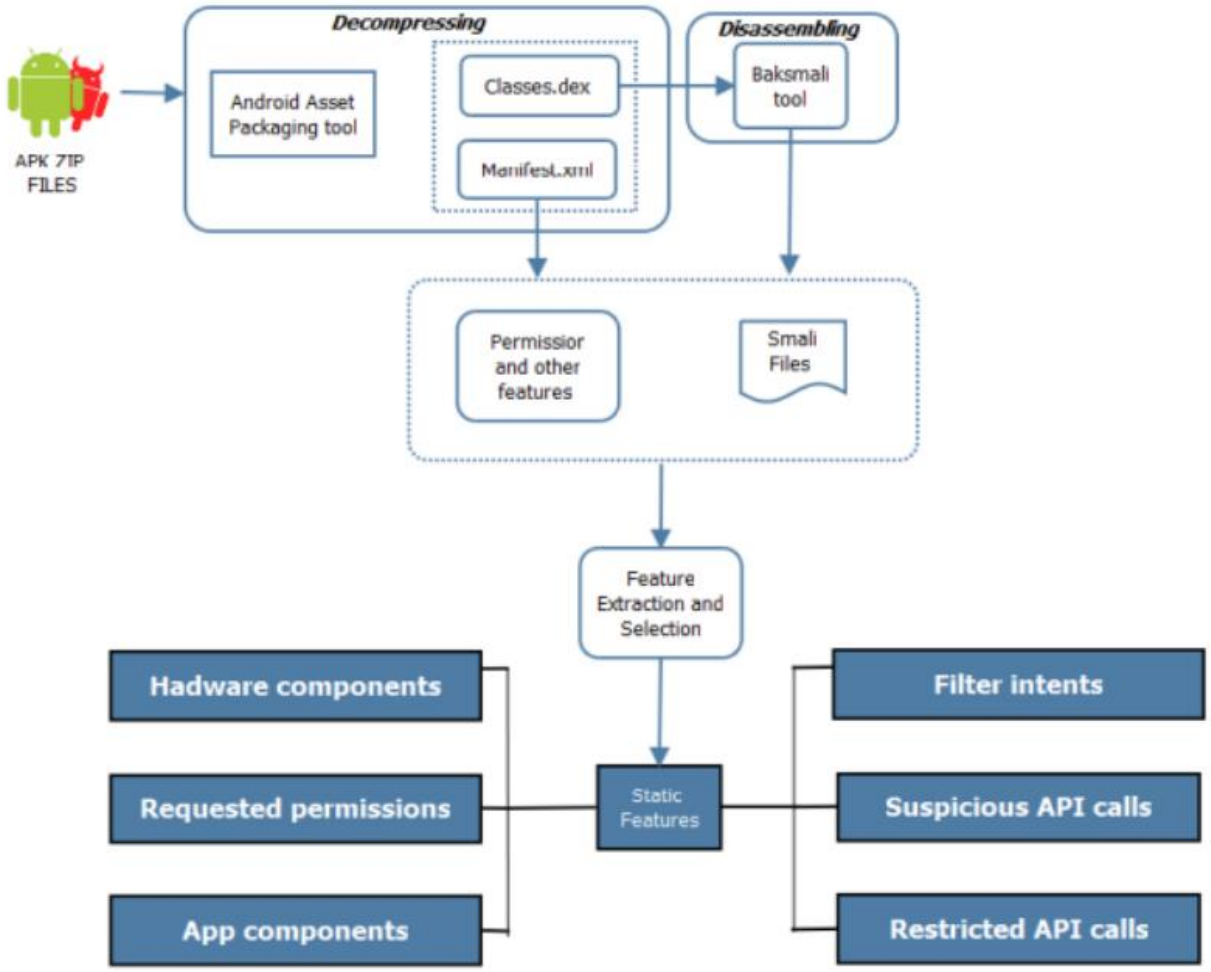


Figure 2. Static feature extraction process

- ✓ Dynamic analysis to observe the real-time behavior of the phone, observe the dynamic behavior and characteristics of the application. From this point of view, the dynamic behavior of malware activities is analyzed, and the simulator (Android virtual device) is used to extract dynamic features, such as API calls, events/actions.

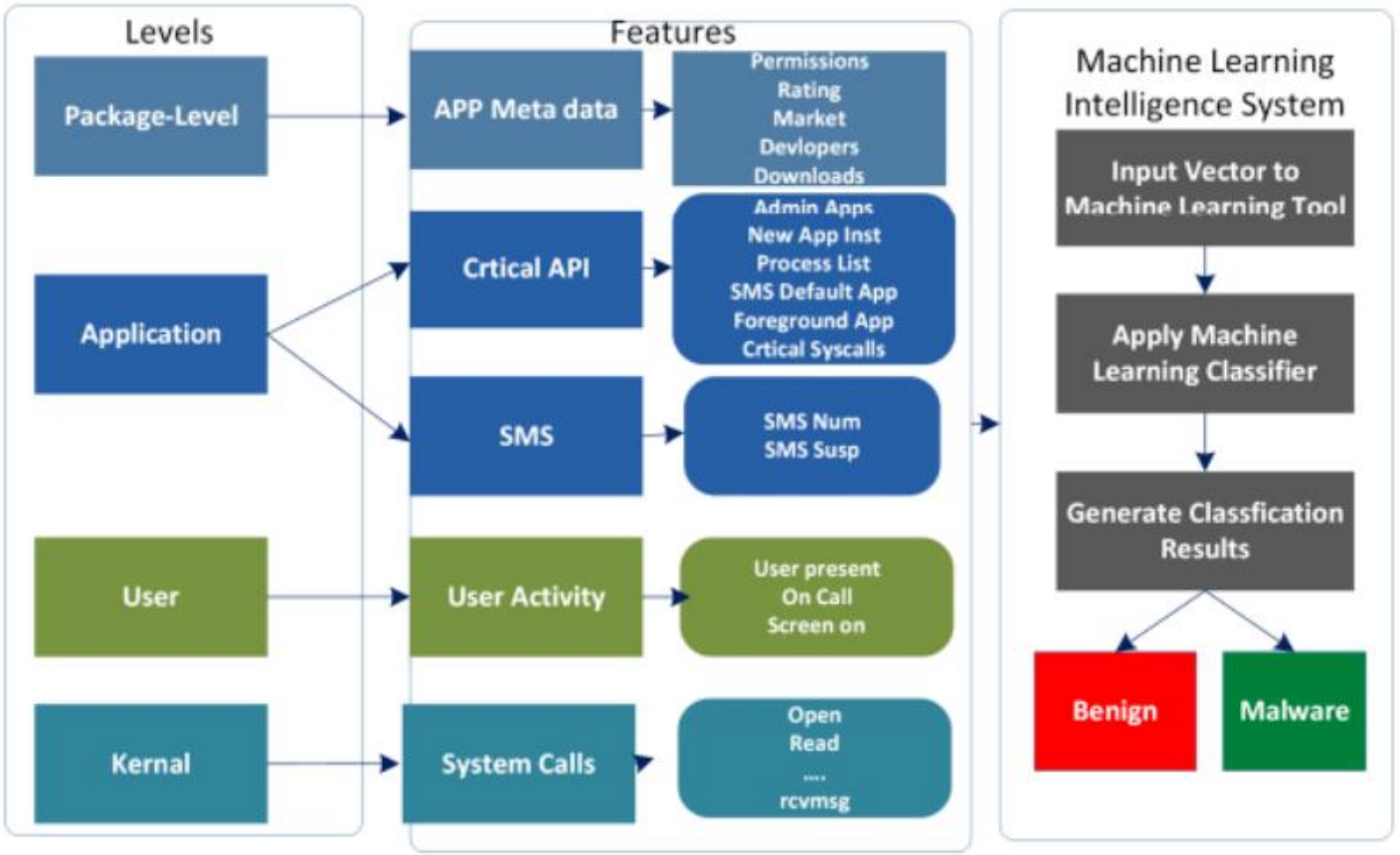


Figure 3. Traditional dynamic analysis features extraction and classification process

- ✓ Hybrid analysis combines static and dynamic analysis. Static features are extracted without executing the application. In contrast, dynamic features are extracted through simulators or on real devices, which is time-consuming and laborious.
- ✓ The author designs a blockchain-based framework that distributes resources equally to all users. Combining static and dynamic analysis methods with blockchain and deep learning, the author improves the detection rate and overcomes the weakness of machine learning. In addition, our approach is to distribute malware information in the blockchain network to notify benign and malicious Android applications at the time of installation.

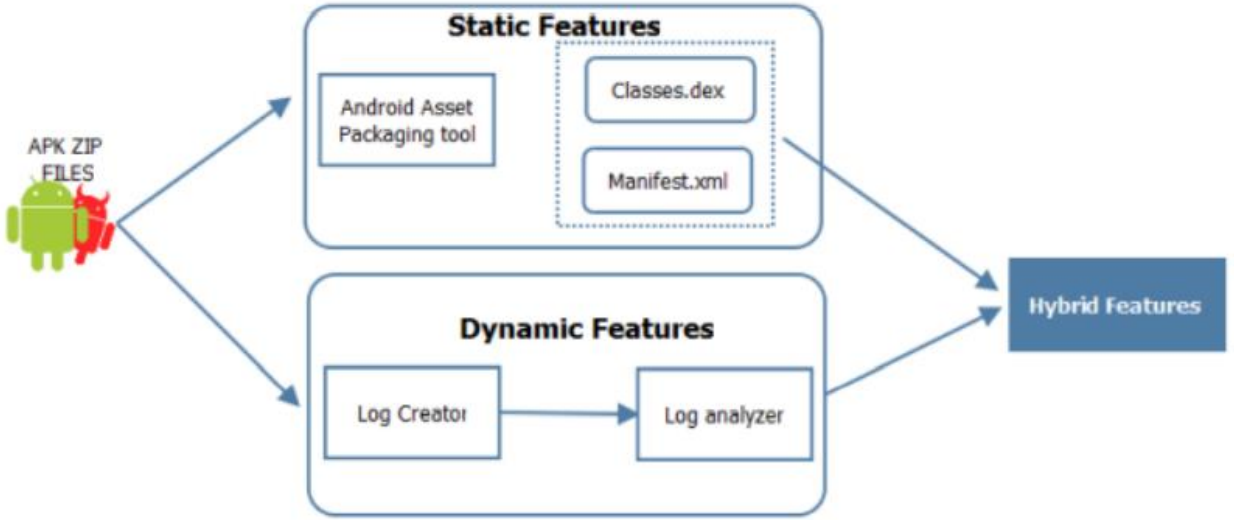


Figure 4. Hybrid feature extraction process

System Model

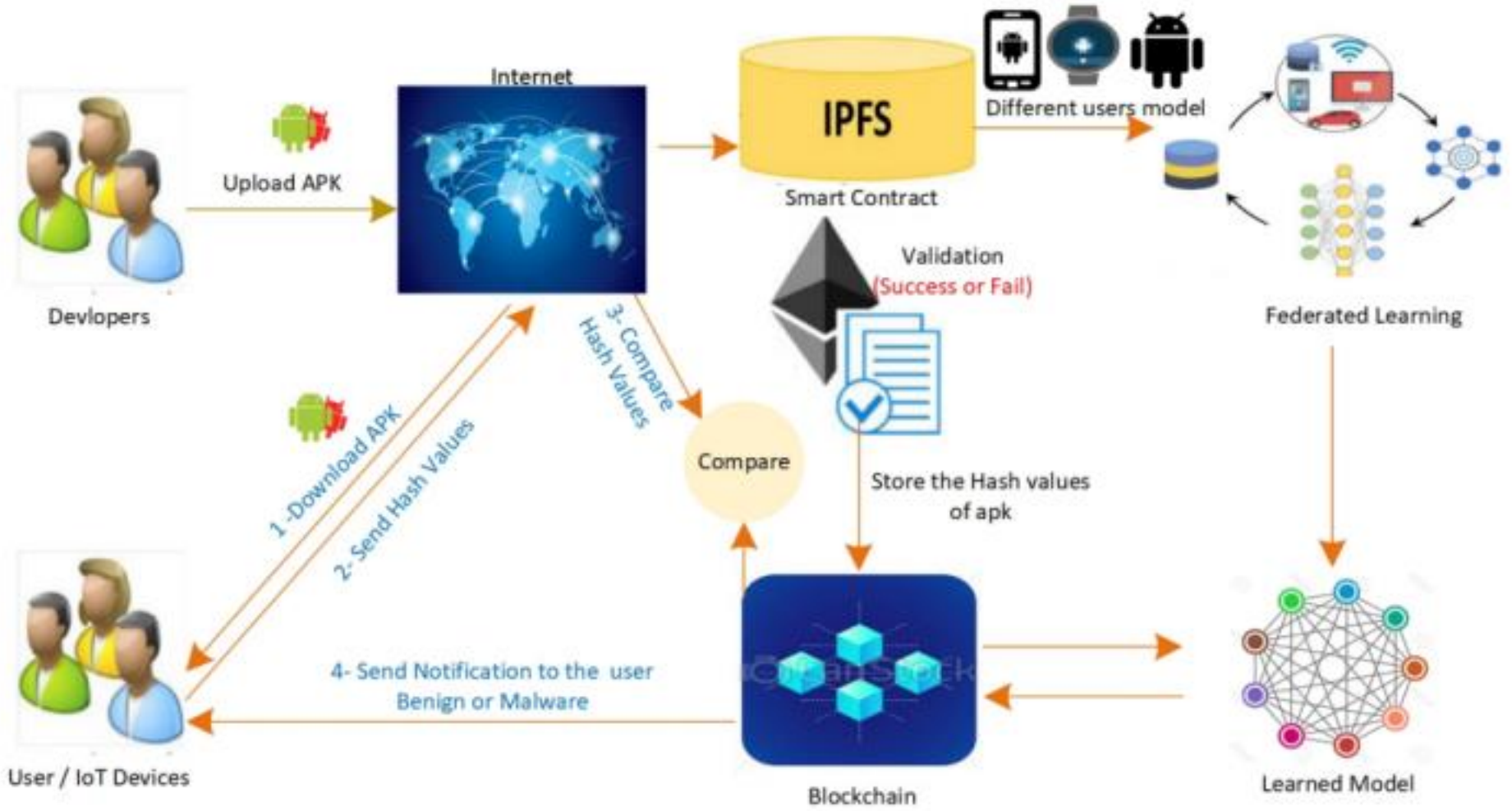


Figure 5. Proposed framework based on Federated learning and blockchain

Deep learning local model

- ✓ Figure 7 shows the overall architecture of the deep learning model for both static and dynamic analysis.
- ✓ Feature selection for mixed malware detection: We utilize the feature importance attribute of the model. Feature importance gives a score for each feature whose data is between 0 and 1. The higher the score, the more important or relevant the characteristics of the output variable are. This score helps to select the most important features and discard the least important features to build the model. Feature Importance is a built-in class that comes with a tree-based classifier. information gain (IG) was used to select important features with high scores and effectively classify the data.

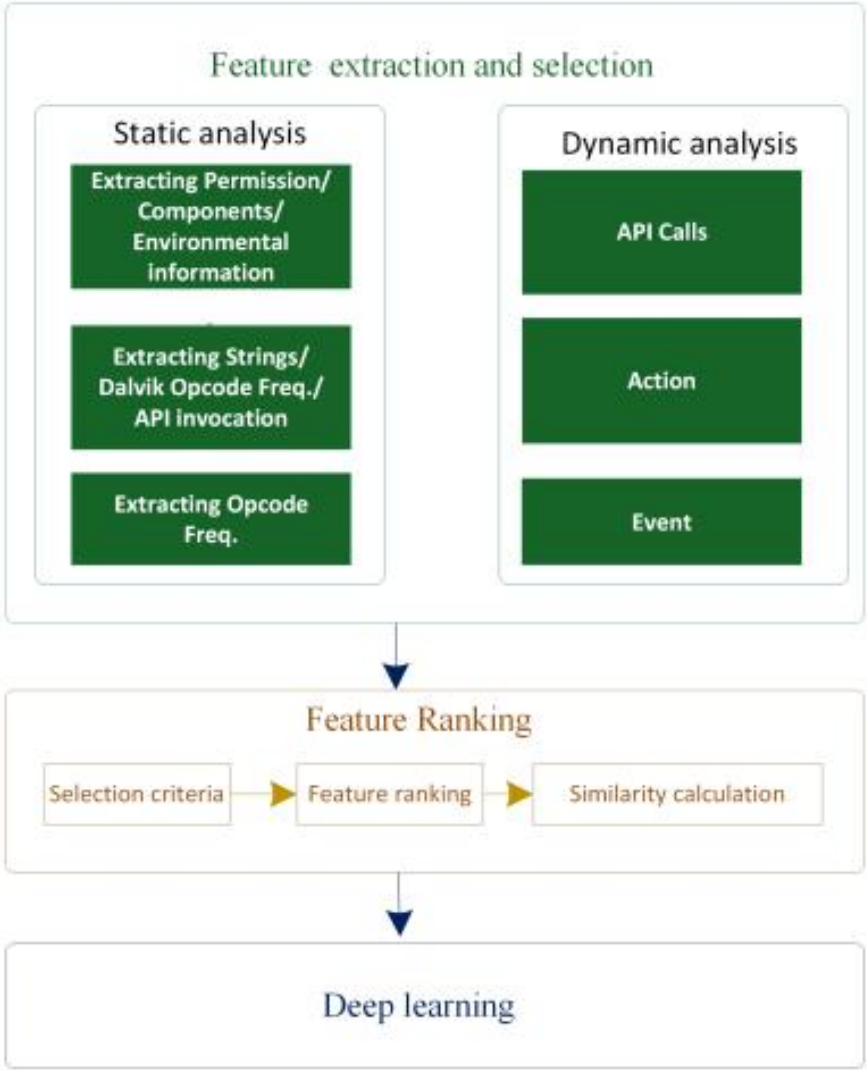


Figure 7. Static and dynamic analysis local deep learning model

Deep learning local model

Feature importance gives a score for each feature whose data is between 0 and 1. The higher the score, the more important or relevant the characteristics of the output variable are. This score helps to select the most important features and discard the least important features to build the model. Feature Importance is a built-in class that comes with a tree-based classifier. information gain (IG) was used to select important features with high scores and effectively classify the data

$$Gain(A) = Info(D) - Info_A(D) \quad (1)$$

$$info(D) = -\frac{pos}{total}\log_2\frac{pos}{total} - \frac{neg}{total}\log_2\frac{neg}{total} \quad (2)$$

$$\text{InfoGainRatio}\ (A) = \frac{Gain(A)}{Info(D)} \quad (3)$$

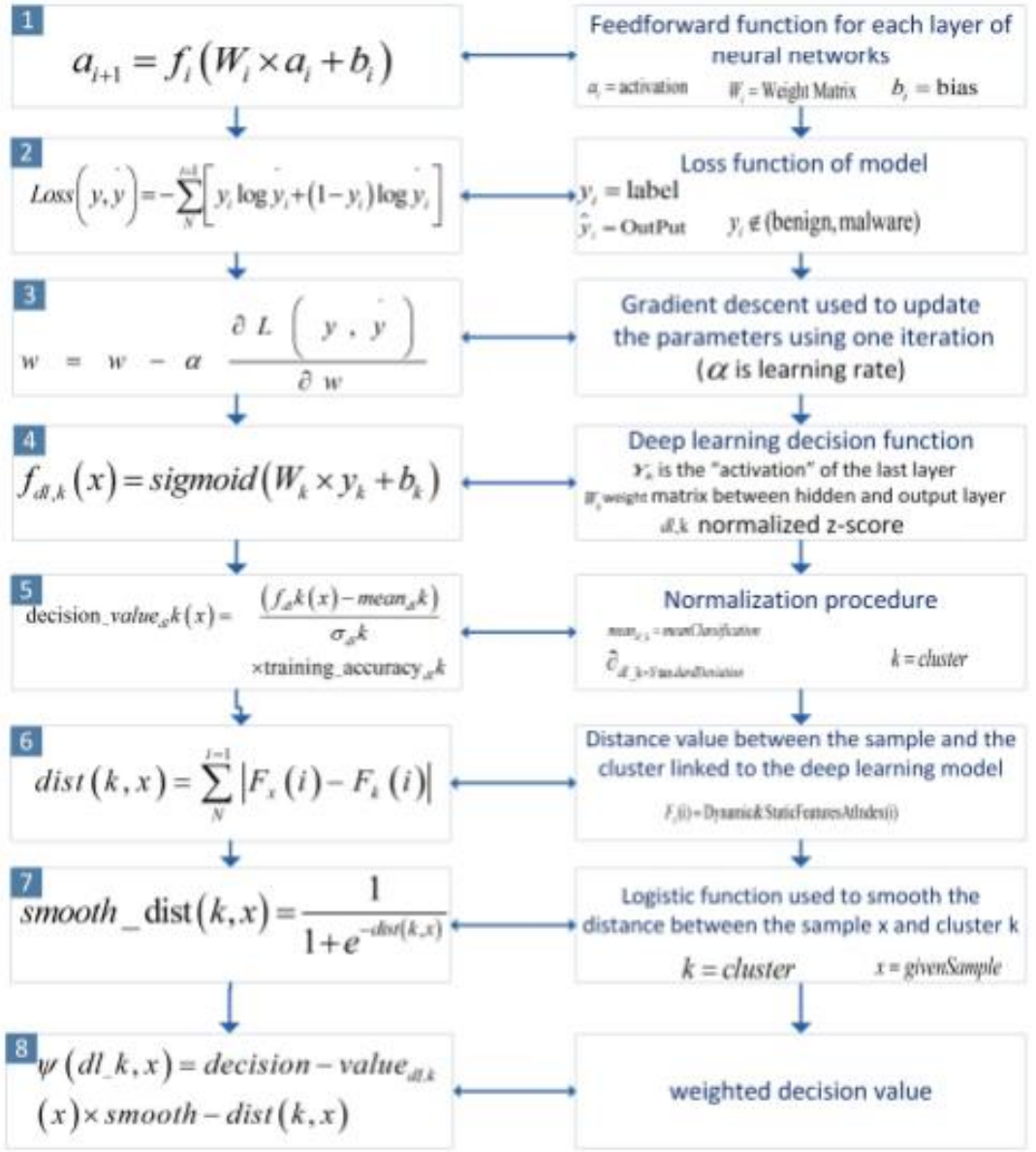
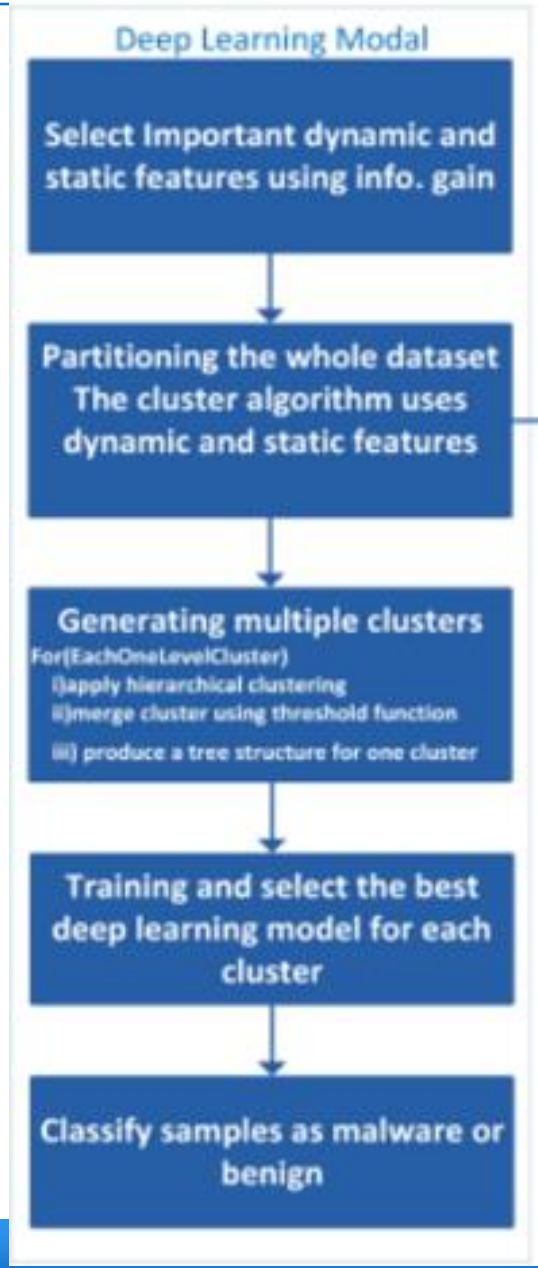


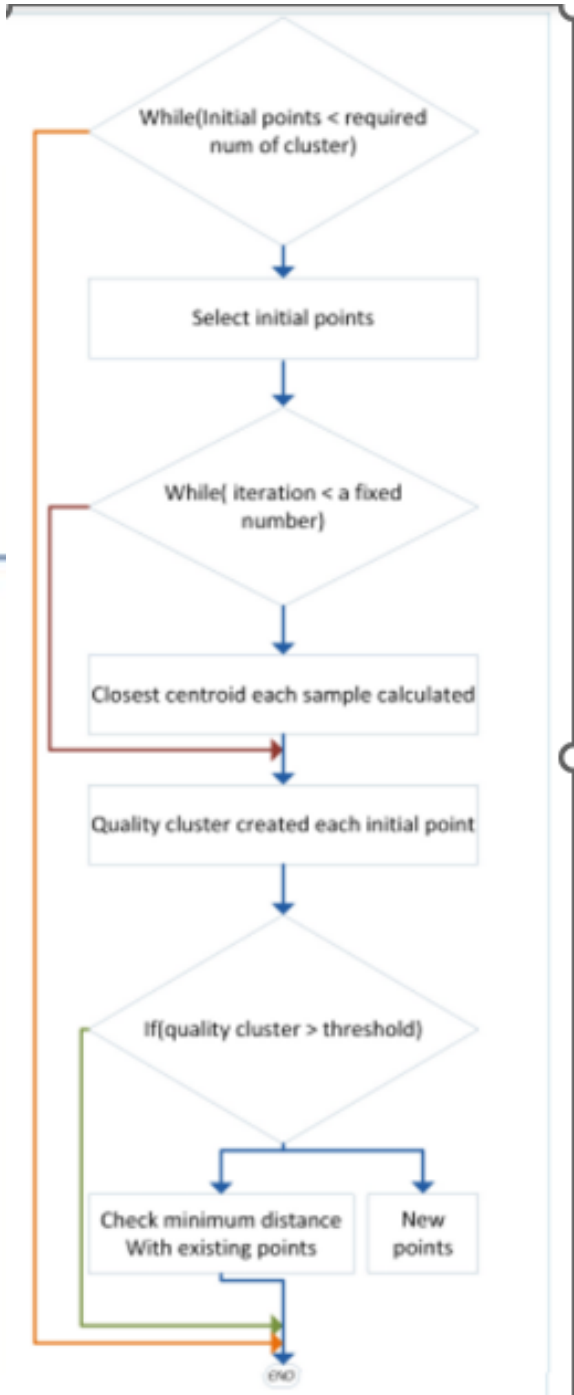
Figure 8. Deep learning model training steps.

Deep learning local model

- ✓ First, the information gain function is used to select important features for static and dynamic analysis.
- ✓ In the second stage, the data sets are moved into different clusters and unique data distributions are calculated.
- ✓ In the third stage, for the large number of features, multiple clusters are generated as the subtrees of each tree cluster.
- ✓ In the fourth stage, the optimal deep learning classifier is selected to assign malware and benign software from the unique data distribution of each cluster.

However, the proposed deep learning model performs static and dynamic analysis for each cluster with each different feature. Using multiple deep learning models during the training phase can reduce time and provide better accuracy. Finally, the proposed model classifies malware and benign software. Using multiple deep learning models during the training phase can reduce time and provide better efficiency. Finally, the proposed model classifies malware and benign software.





✓ The locally trained model is aggregated with new information on the characteristics of the latest feature application, and the latest schema in IPFS is updated to track new harmful information applications. The process of combining blockchain and federated learning technologies is shown in Algorithm 1. The weight of the layer is calculated by training the neural network with (i) forward propagation and (ii) backward propagation.

$$F = (x_i, y_i); i \epsilon I$$

$$loss = \frac{1}{F} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in F} l\left(\mathbf{y}_i, f\left(\mathbf{x}_i, \mathbf{w}\right)\right)$$

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \nabla_{\mathbf{w}} L\left(F^t, \mathbf{w}^t\right) \quad (4)$$

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \frac{\sum_{v \in V} \nabla_{\mathbf{w}} L\left(F_v^t, \mathbf{w}^t\right)}{|V|} \quad (5)$$

The Federated learning model identifies malware applications by calculating gradients and sending updated weights to the global blockchain network. The smart contract shares the updated results of the aggregation. In addition, smart contracts can identify harmful apps when users download them.

Algorithm 1: Aggregate deep learning weights from the blockchain network

MD ← MobileDevices ;
$\{F_n\}^{n \in [N]}$ ← Malware Features ;
w^0 ← global weights ;
$L(w, x)$ ← Loss ;
I ← iteration ;
θ ← clip bound ;
for $i \in [I]$ **do**
  for $md \in [MD]$ **do**
    sample malware and benign feature data set with probability $\frac{|f_{md}^i|}{|f_{md}|}$;
  end
  for $\mathbf{x} \in F_h^t$ **do**
    $gf_{md}^i(\mathbf{x}) \leftarrow \nabla_{\mathbf{w}^i} L\left(\mathbf{w}^i, \mathbf{x}\right)$;
    $gf_{md}^i(\mathbf{x}) \leftarrow gd_{md}^i(\mathbf{x}) / \max\left(1, \frac{\|gd_i^2(\mathbf{x})\|}{\theta}\right)$;
    retrieves the weights or global model from permissioned blockchain ;
  end
  $gf_{md}^i \leftarrow \sum_{\mathbf{x} \in D_{md}^i} gd_{md}^i(\mathbf{x}) + \mathcal{MD}\left(0, \frac{\theta^2 \rho^2}{MD}\right)$;
  executes IPFS model to aggregation and obtain updated the IPFS model ;
  add the parameters of model as a transaction ;
end
$gf_{md}^i \leftarrow \frac{1}{MD}\left(\sum_{n \in [md]} gf_{md}^i\right)$;
$\mathbf{w}^{i+1} \leftarrow \mathbf{w}^i - \eta \cdot gd^i$;
retrieves the current updated weights from IPFS, and aggregates the weights;
broadcasts new malware information to other delegates for verification, and collects all transactions into a new block;
appends the block including the global model to the permissioned blockchain;

Storing information

Hashes of Android applications with malicious and benign characteristics (static and dynamic) are stored in a blockchain distributed database.

The first part of the bulk stores the version number of the application, Markle root, hash values for all applications, and so on. The second part of the block data stores all static and dynamic characteristics, such as suspicious APIs, permissions, events, calls, and so on.

Table I
BLOCKCHAIN ATTRIBUTES

Keywords	Size	Definition
Pre- Hash	32 bytes	preceding block hash value
Version number	4 bytes	track the protocol or software updates
Timestamp	5 bytes	records the time a block
Transaction_count	15 bytes	number of malware results in the current block
Merkle root	32 bytes	it calculate the malicious codes which detected by block
Nonce	15 byte	randomly recognized as a formal block

Figure 10. Blockchain data-store technique for multi-features of Android malware

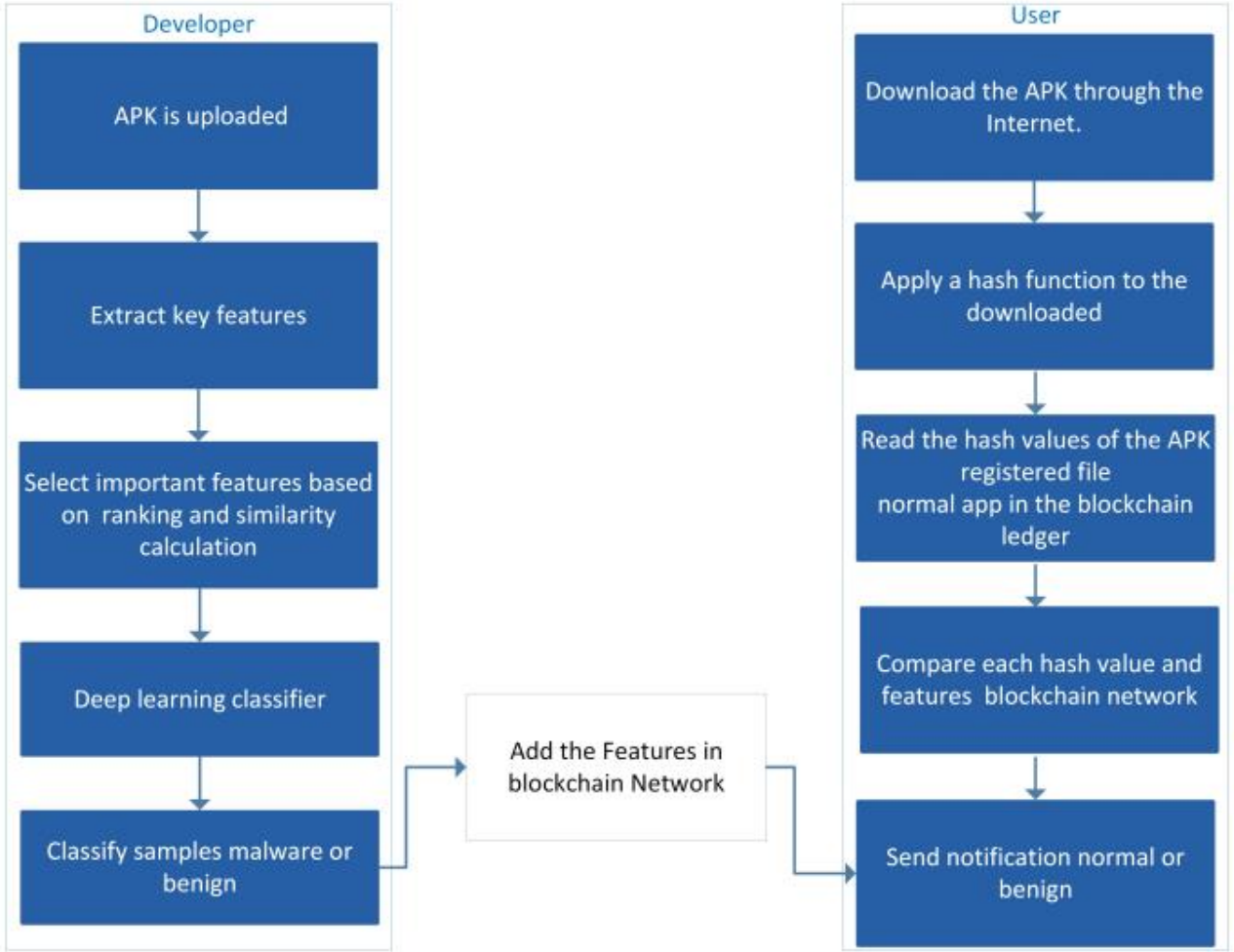


Figure 6. Flowchart of user and developer for Android malware detectio using blockchain

Algorithm 2: Smart contract approvers uploaded apk

```
1  Contract is: WaitForCheckingMalwaree;
2  Devloper is: ReadtToUploadAP;
3  Approva is: WaitingToSucessOrFail;
4  if apkHashCheck(distributedLedger) then
5  |   Contract is: sucessSign;
6  |   Devloper is: sucessProvidedAP;
7  |   Approve = sucessApproval(If app is not Malware);
8  else
9  |   Contract is: denySign;
10 |   Devloper is: denyProvidedAP;
11 |   Approve = denyApproval(If app is Malware);
12 end
```

Algorithm 3: Smart contract approvers download apk

```
1  apk ← DownlodedApp;
2  apkHash ← ApplyHash(apk);
3  Approve is: WaitingToSucessOrFail;
4  if BlockChainLedger(apkHash) then
5  |   Approve = sucessApproval(Malware information
   |    not found);
6  else
7  |   Approve = denyApproval;
8  end
```

✓ Evaluation indicators:

$$TPR = \frac{T_p}{T_p + F_n} \tag{6}$$

$$FPR = \frac{F_p}{F_p + T_n} \tag{7}$$

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{8}$$

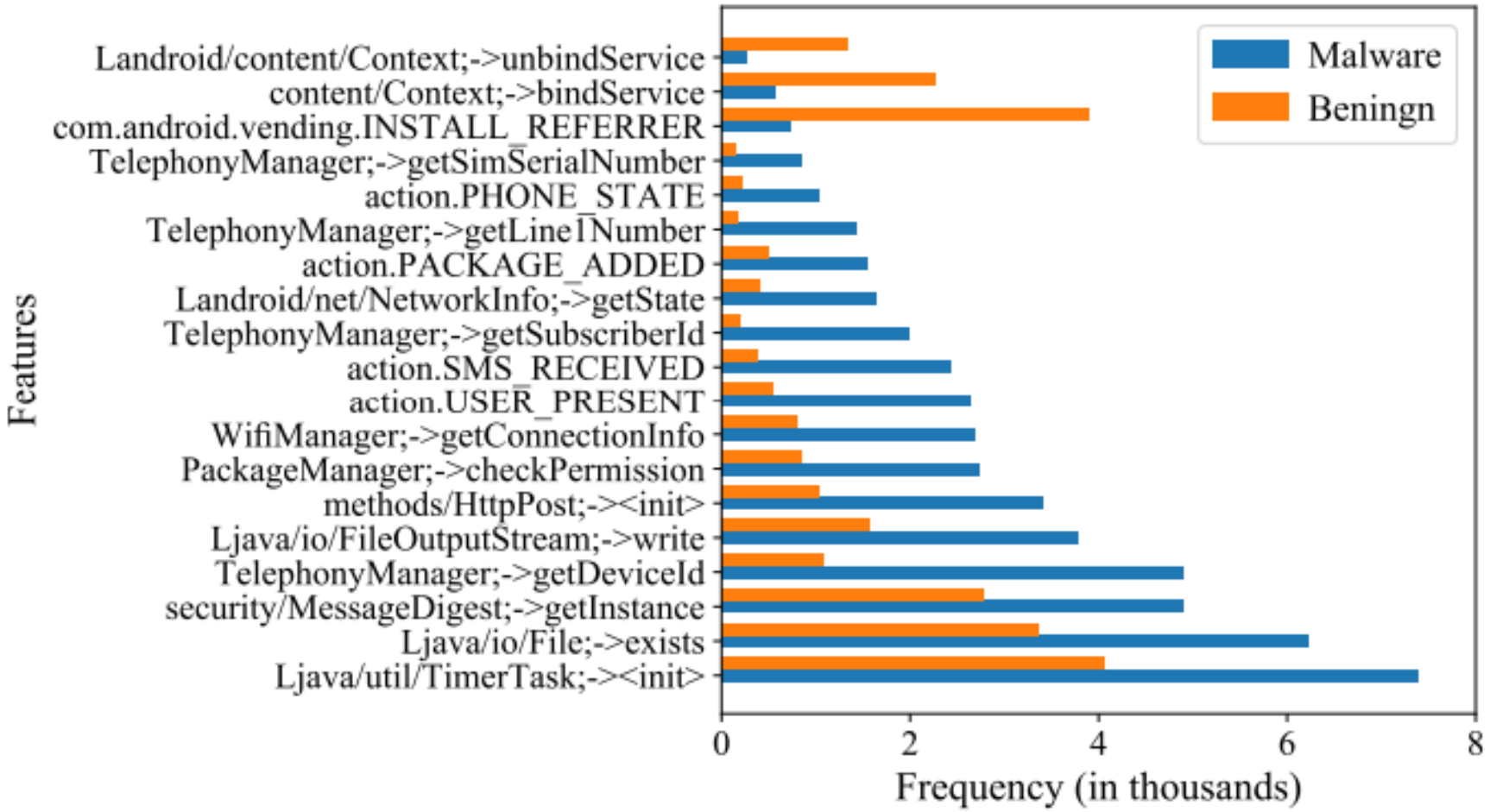


Figure 11. Top ranked info-gain-based apps use the DroidBot (Permission Excluded)

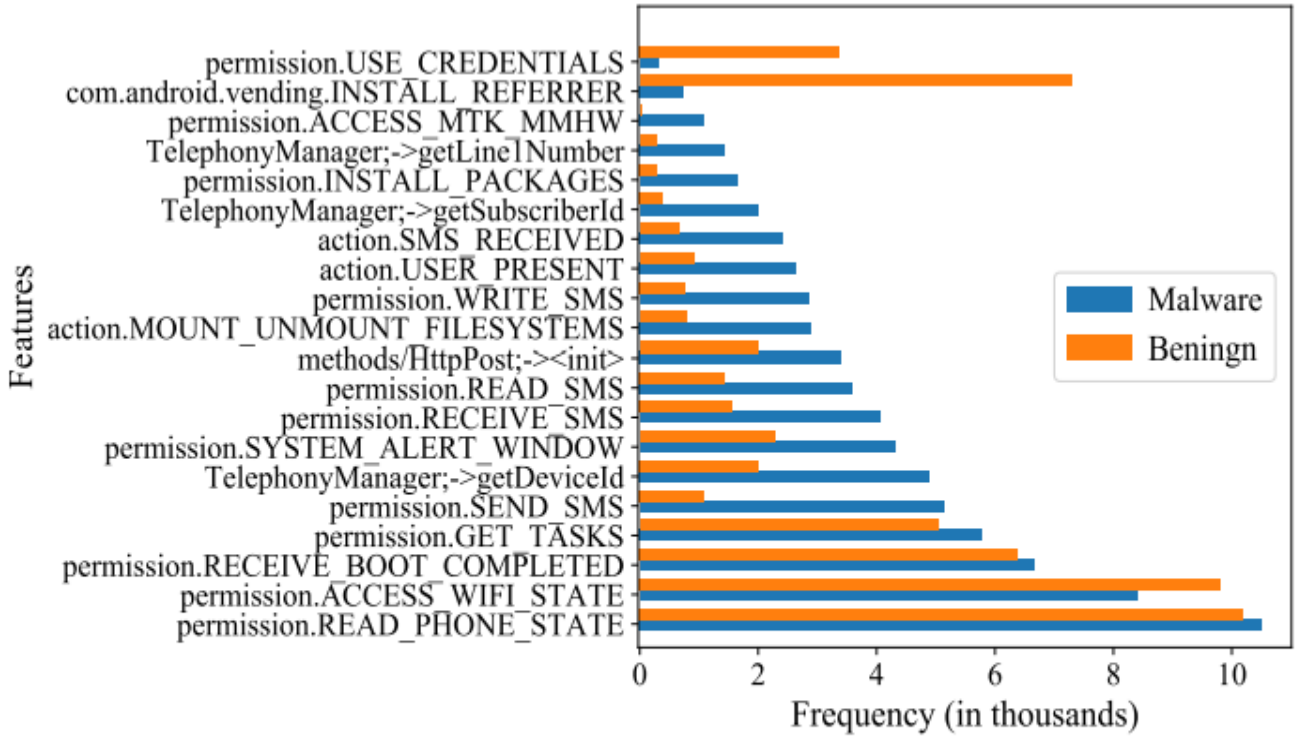


Figure 12. Top ranked info-gain-based apps use the DroidBot (Permission Included)

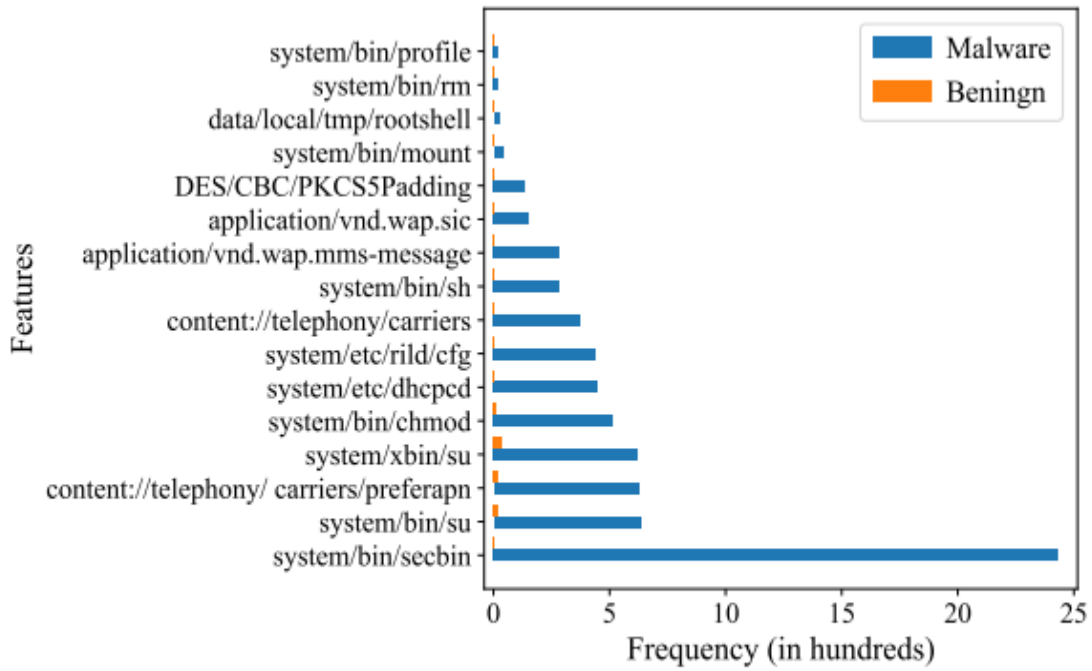


Figure 13. Top ranked info-gain based Process

Because of the same frequency count, use permissions do not help distinguish between benign and malicious software, as shown in Figure 12.

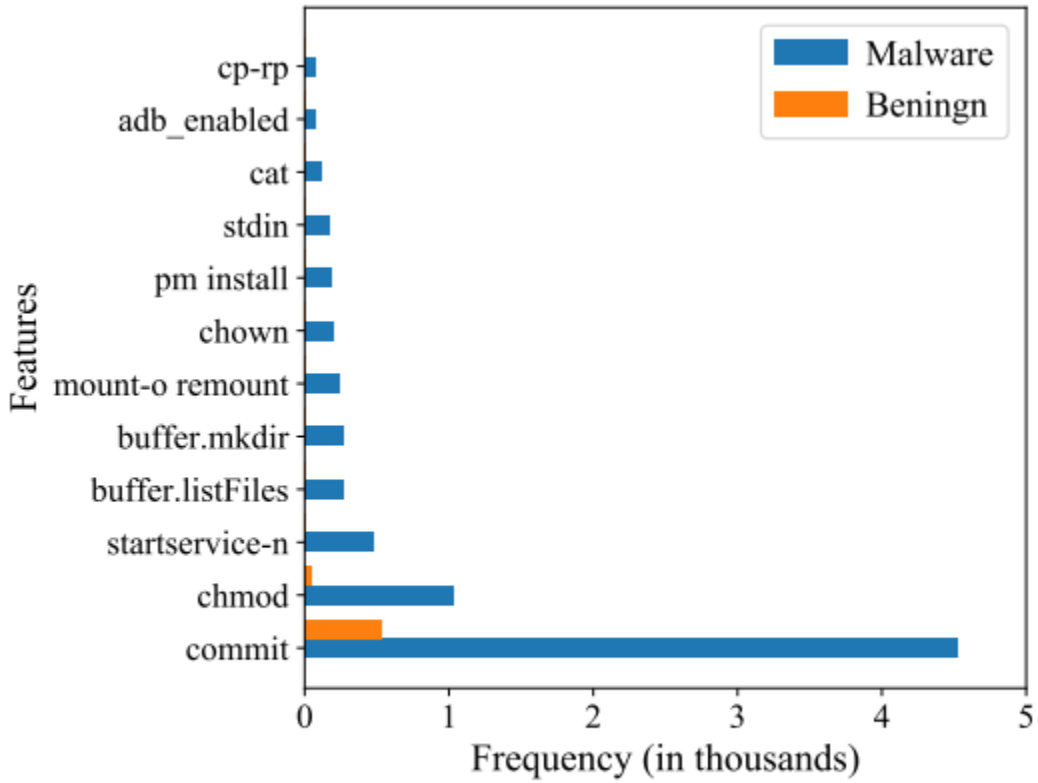


Figure 14. Top ranked info-gain based Intents

There is an inverse relationship between the frequency of API calls and services between benign and malicious applications. Specifically, as shown in Figure 15, the frequency count of benign intent is much lower than that of malware applications.

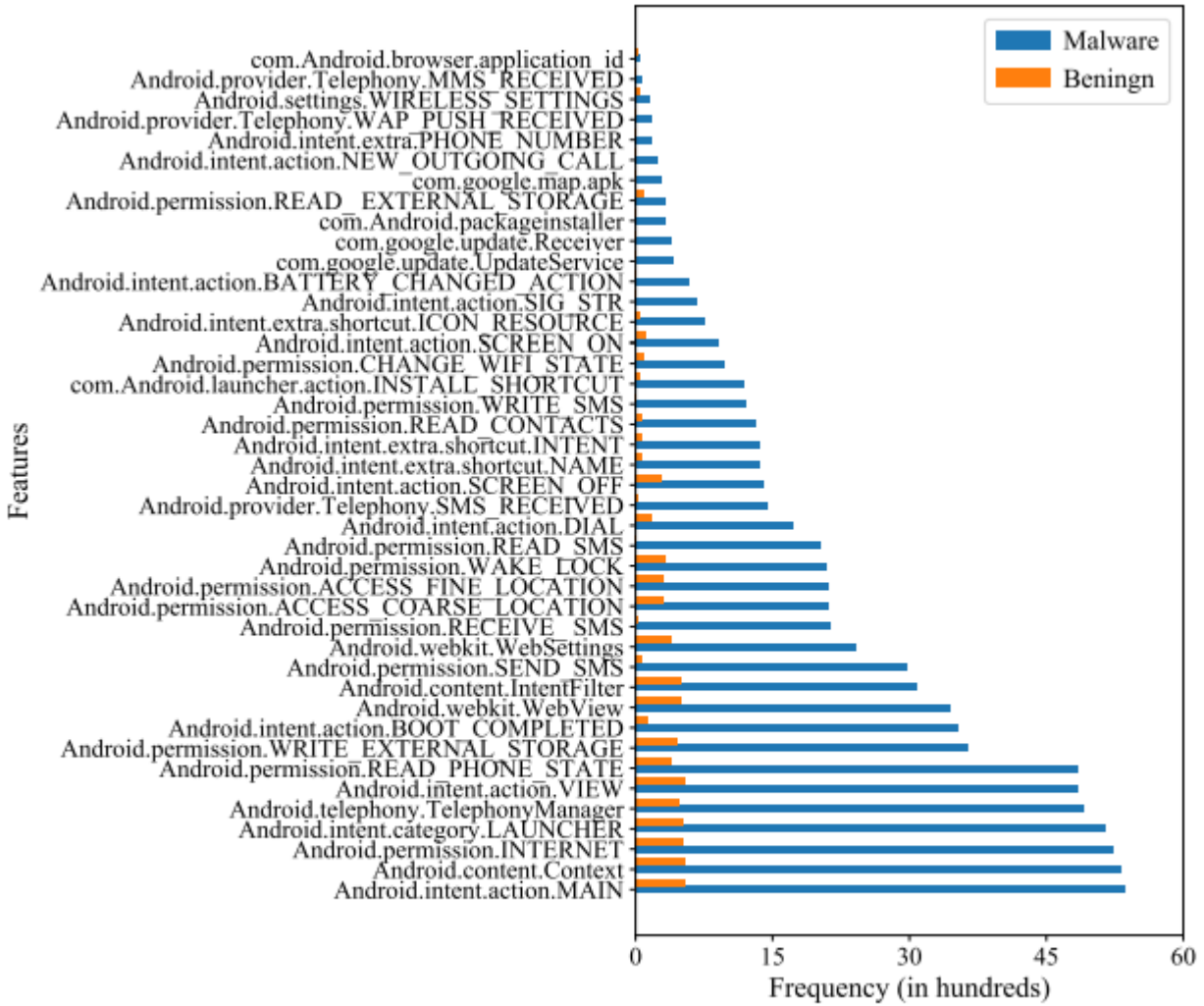


Figure 15. Top ranked info-gain based Intents

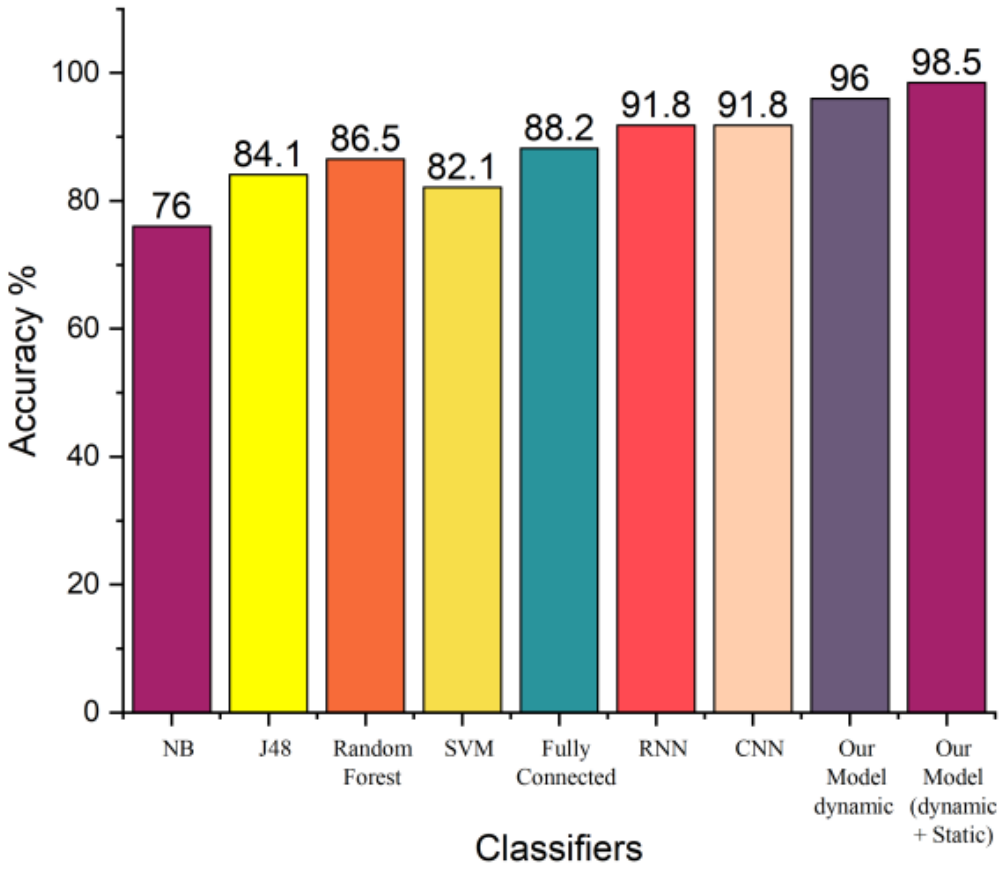


Figure 16. Comparison between machine and deep learning classifiers

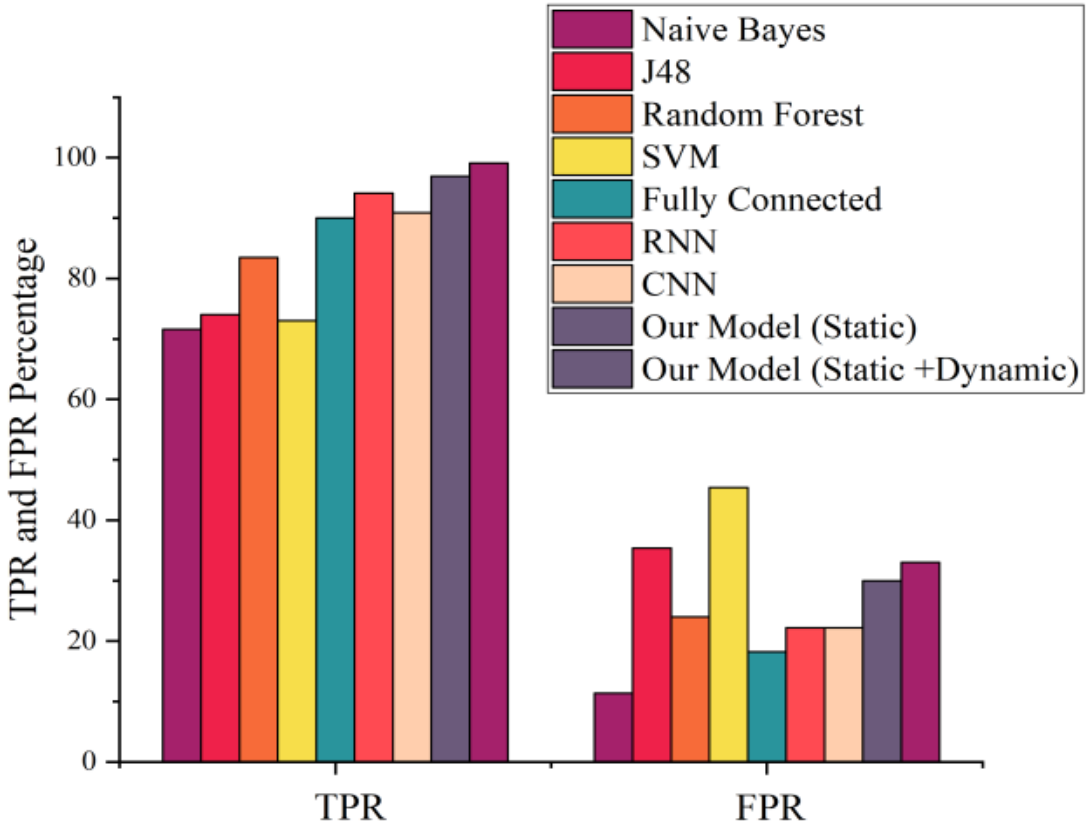


Figure 17. True positive and false positive performance between different classifiers

As can be seen from Figure 16, the proposed method achieves higher TPR and accuracy, which is better than the previous algorithm. However, due to the conflicting characteristic relationships between benign and malicious software, the reported FPR methods are not superior to other methods except SVM and J48.

- ✓ Table 2 shows the performance of the deep learning model under different hidden layer combinations. Table 2 The combination of 200, 200 and 200 neurons applying the two-layer, three-layer and four-layer deep learning model has the best results compared with other layers and neurons.

Table II
PROPOSED DEEP LEARNING MODEL WITH DIFFERENT HIDDEN LAYERS FOR DYNAMIC FEATURES ONLY

No. of lay-ers	No. of Neurons	TPR	FPR	Accuracy	Running time (min:sec)
2	200,200	0.9663	0.337	0.9449	06:31
2	400,400	0.9903	0.2062	0.895044	13:44
3	**200,200,200**	**0.9719**	**0.0728**	**0.9624**	**09:05**
3	400,400,400	0.7219	0.0158	0.8183	20:40
4	200,200,200,200	0.9744	0.0891	0.9508	10:39
4	400,400,400,400	0.9622	0.115	0.9339	29:28

Table III
PROPOSED DEEP LEARNING MODEL WITH DIFFERENT HIDDEN LAYERS FOR STATIC AND DYNAMIC FEATURES

No. of lay-ers	No. of Neurons	TPR	FPR	Accuracy	Running time (min:sec)
2	200,200	0.9661	0.1229	0.9332	14:51
2	400,400	0.972	0.0918	0.9484	30:50
3	**200,200,200**	**0.9956**	**0.033**	**0.985**	**17:21**
3	400,400,400	0.9764	0.0834	0.9543	38:16
4	200,200,200,200	0.9757	0.0793	0.955	20:05
4	400,400,400,400	0.9717	0.0907	0.9486	43:52

- ✓ Table IV compares the time of different deep learning models. The experimental results show that this model not only reduces the computing time, but also achieves the detection performance of Android IoT devices.
- ✓ In Table V, we classify Android malware by blockchain based on joint learning and joint learning models.
- ✓ Federated learning takes less time than training local models. It takes 0.93 milliseconds for the client to send the model from the client to the server. Therefore, the proposed framework has less communication time. Blockchain and deep learning models aggregate different characteristics, namely permissions, intentions, dynamic characteristics, and static characteristics, which perform better than other previous approaches.

Table IV

TIME COMPARISON OF DEEP LEARNING MODEL CONSTRUCTION

No. of layers	No. of Neurons	TPR	Time
3	200,200,200	Fully Connected	140
3	200,200,200	RNN	135
3	200,200,200	CNN	120
3	200,200,200	Our Proposed (Static)	96
3	200,200,200	Our Proposed (Dynamic)	99

Table V

PERFORMANCE OF FEDERATED AND BLOCKCHAIN-FEDERATED METHODS

Train on	TPR	FPR	Accuracy	Time in seconds
Local user 1	97.41	17.92	93.95	189.05
Local user 2	96.15	13.71	93.59	
Local user 3	95.70	13.13	93.34	
Local user 4	95.66	14.22	93.41	
Local user 5	96.47	14.70	93.64	
Federated	97.34	12.41	95.35	0.052
Blockchain-Federated	98.64	13.21	98.05	times

- On the Android platform, there are many clients using many kinds of services, so we tested the classification performance based on the number of clients. Each user selects different 10,000 feature sets to randomly compose the training data set, and trains local models with various features. According to Table 6, as the number of clients increases, the accuracy increases. Therefore, the more users you have, the better your model will perform.
- There are many different characteristics in the Android malware dataset to detect malware, so we categorize the different characteristics set in Table VII and compare them with Federated Learning.

Table VI
PERFORMANCE OF DIFFERENT NUMBER OF CLIENTS

Train on	TPR	FPR	Accuracy
2 clients	96.45	13.67	96.99
5 clients	97.54	12.49	97.45
10 clients	97.43	12.18	97.74
15 clients	97.72	13.78	97.98
30 clients	97.43	12.41	97.98
40 clients	96.54	11.89	97.45

Table VII
PERFORMANCE OF DIFFERENT TYPES OF FEATURES

Feature	TPR	FPR	Accuracy
API calls	95.12	20.34	91.41
Permission	85.90	31.14	81.43
Intents	92.34	19.24	92.12
Federated	98.64	20.75	94.42
Federated Blockchain	99.10	17.87	98.62

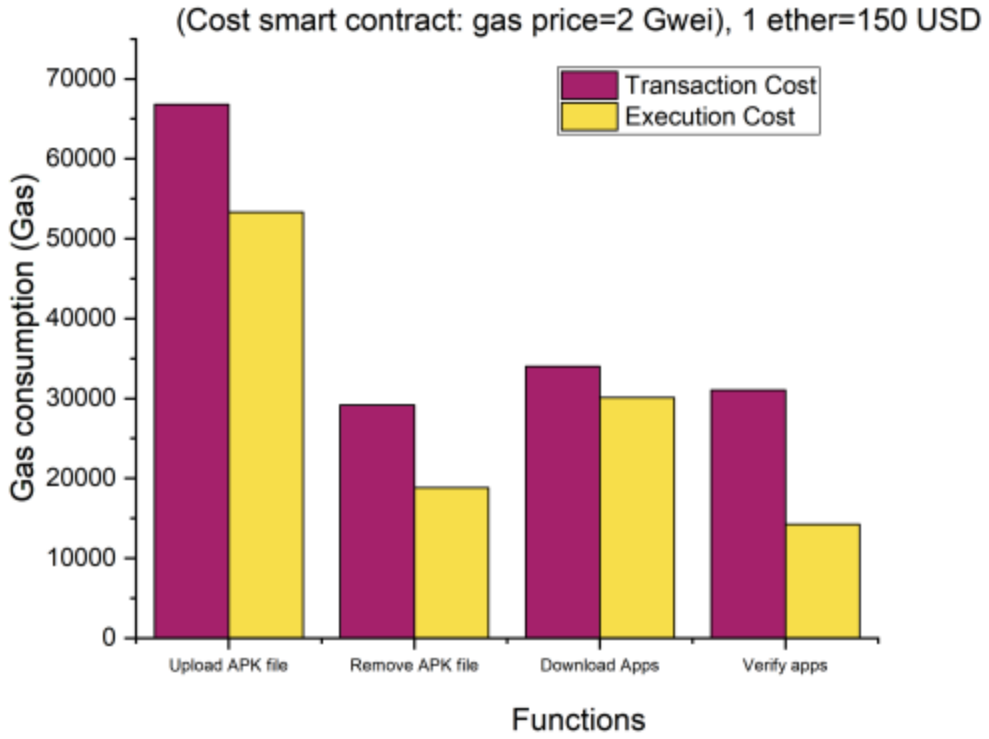


Figure 18. Gas consumption for downloading and uploading file

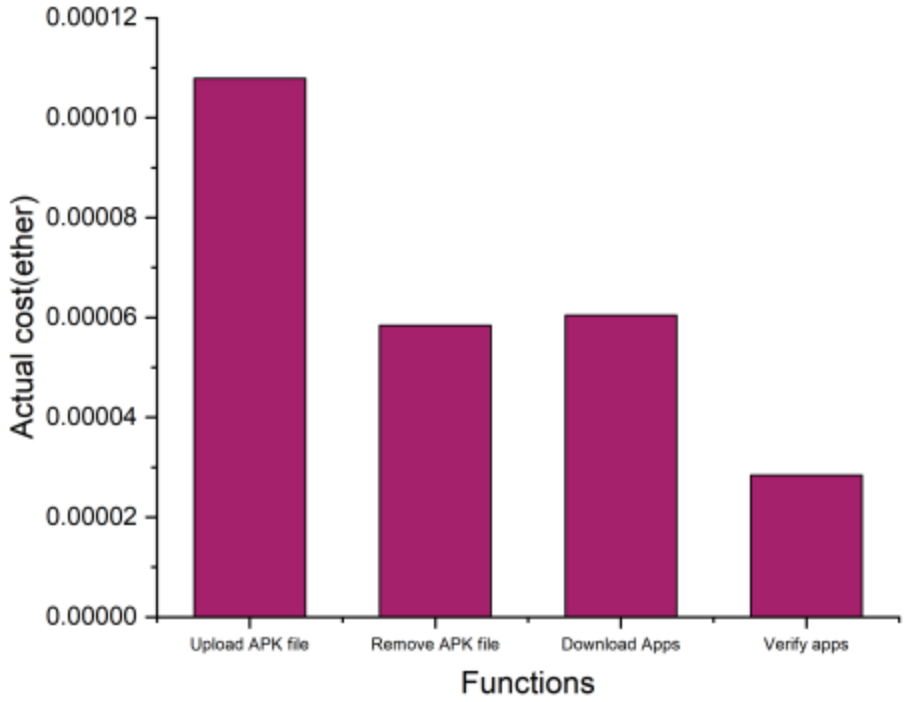


Figure 19. Actual cost for downloading and uploading file

The author implemented the Ethereum smart contract using the Remix IDE. All roles are tested to make sure the smart contract works.

Computing power and average transaction

- ✓ With the addition of training labels, the predictive performance of the deep learning model is improved.
- ✓ The combination of blockchain and deep neural networks improves performance in terms of reducing the computing cost of neural networks

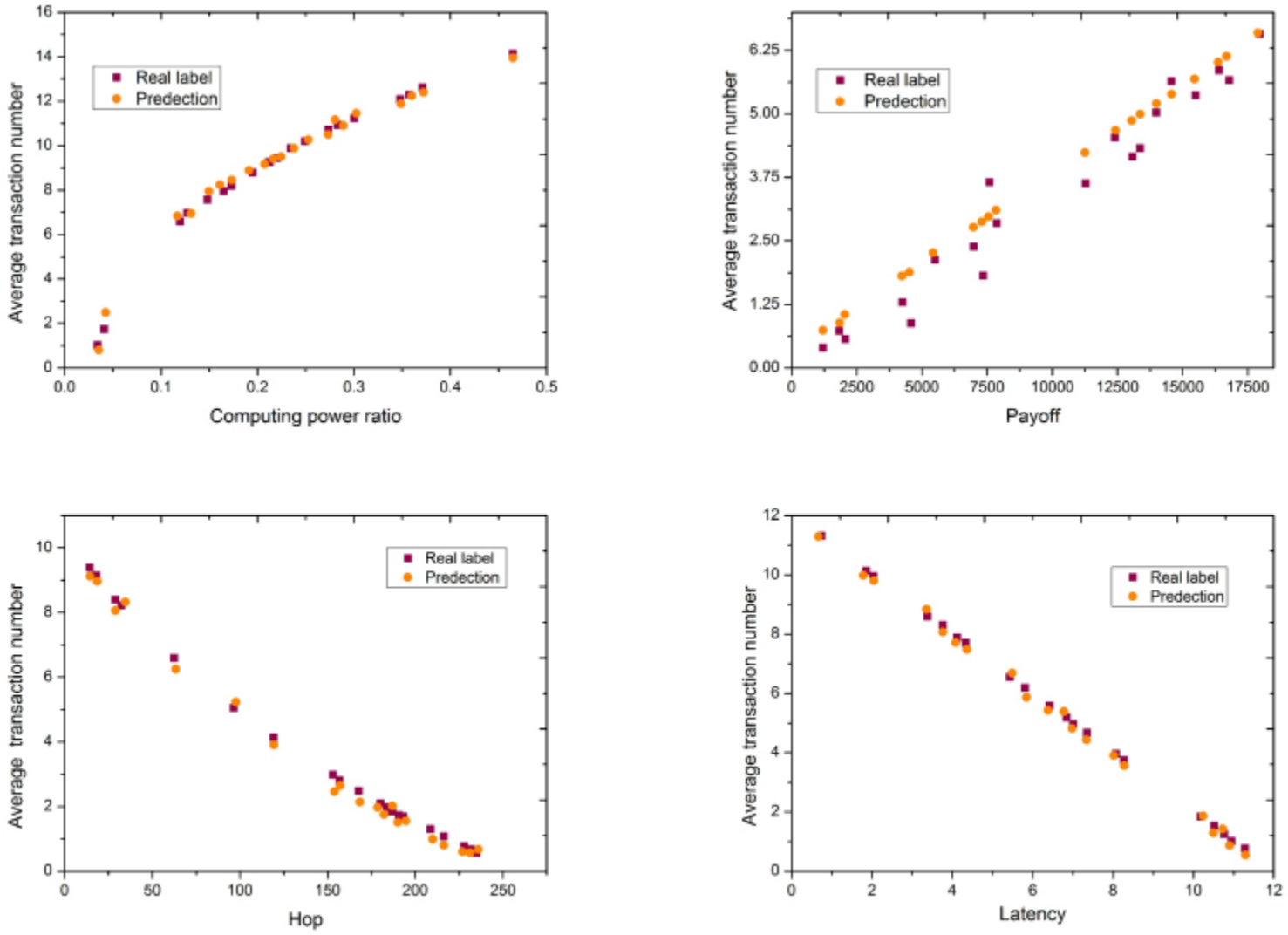


Figure 20. Correlation between the computing power ratio and average transaction

Authors	Algorithm	Capacity for feature diver-sity	Accuracy	F-measure
OURs	Proposed	High	96%	0.98
[13]	DNN/RNN	medium	90%	NA
[80]	CNN	low	90%	NA
[50]	Multi-Layer Perception	low	89%	0.89
[43]	KMNN/ ANN/ FNN	High	90 %	NA
[54]	RNN and LSTM	low	96%	NA
[12]	DNN	High	93.9	NA
[81]	Bayesian	low	92%	NA
[82]	SVM	low	NA	0.98
[64]	Graph Based	NA	95.4%	NA

Table VIII

PERFORMANCE COMPARISIN WITH OTHER STATE OF THE ART APPROACHES

Primitive	**Block Verify [38]**	**Sigma Ledger [39]**	**Stop TheFake [36]**	**This Work**
Blockchain	Private	Private	Private	Private
Target	Goods	Goods	Picture, Video	Android APK
Function	Detect, Identify	Tag, Detect	Detect, Record	Detect, Identify
Smart Contract	Product Label	QRCode, RFID	Copyright, Catalog	Hash, Feature of APK

Table IX

COMPARE WITH OTHER BLOCKCHAIN TECHNIQUES

This paper presents a new approach that integrates blockchain and a multi-level deep learning model for the detection of malware activity in a real-time environment, specifically for Android IoT devices.

1) The developer creates a malware
2) Multi-layer deep learning model Distributed malware features into different clusters, and selected the best deep learning model for each cluster.
3) Make decisions by analyzing data already stored previously in the blockchain's distributed ledger and storing new features of malware activity in the blockchain.
4) Finally, blockchain smart contracts provide notifications (of malware) to users during the upload or download process to verify Android applications. To enable better security for malware detection on IoT devices in a real-time environment, millions of Android app signatures (malware and benign) are stored in a blockchain database.

Therefore, a multi-layer deep learning model is designed for malicious software and benign features that receive a large number of malicious applications in Android IoT devices. The model supports multi-level clustering with a single data distribution. Smart contracts verify malicious apps, and upload and download Android apps over the network. It can approve or reject harmful uploads and downloads of Android apps. This model can effectively identify malicious software and provide higher security for network security.

THANKS

서울과학기술대학교 컴퓨터공학과 진호천