DLINE JOURNALS

Journal of Digital Information Management

ISSN: 0972-7272

JDIM 2025: 23 (1)

An Android Malware Detection Method Based on MLSTM

Yi Liu[1, 2], Md Gapar Md Johar[1, *], Jacquline Tham[2]
[1]Postgrduate Center, Management and Science University
Shah Alam 40100, Malaysia

[2]School of Information Engineering
Gongqing Institute of Science and Technology
Jiujiang 332020, China
mdgapar@126.com

ABSTRACT

With the popularity of smartphones and mobile applications, the threat of Android malware is increasingly serious. The analysis and behaviour modelling of Android malware features is studied to realize the efficient and accurate detection of Android malware, and an Android malware detection method combining mean aggregator and long-term and short-term memory is proposed. The results show that the improved system detection time is relatively stable regardless of the number of samples. The average detection time of the improved and unimproved systems is 0.274 s and 0.336 s, respectively, and the improved detection efficiency of the improved system is more prominent. The highest improvement rate of the enhanced system reached 18.2%. Compared with other models, the average absolute error and root mean square error were the smallest, with 3.84 and 6.26, respectively, indicating that the detection performance of the improved model is the best. With permission features and third-party library features, the accuracy of the enhanced model was 98.89% and 92.65%, and the recall rate was 99.24% and 99.09%, respectively. The improved model detection performance is good, and the robustness and stability are enhanced. Applied to actual Android devices, it can improve the security and privacy protection level of user data. This method ensures enhanced efficiency and stability and provides a certain reference direction for Android malware detection.

Subject Categories and Descriptors: [C.2 COMPUTER-COMMUNICATION NETWORKS]; *Security and protection:* [B.4.1] Data Communications Devices; [*B.4.2*] *Input/Output Devices;* [*I.5*] *PATTERN RECOGNITION Neural nets*

General Terms: Malware detection, Communication Security, Protection, Android phones, Neural Networks

Received: 18 August 2024, Revised 3 November 2024, Accepted 15 November 2024

Keywords: Android Malware Detection, Mean Aggregator, Lstm, Security, Graphsage

Review Metrics: 0/6; Review Score: 5.03; Inter-reviewer Consistency: 86.2%

DOI: https://doi.org/10.6025/jdim/2025/23/1/11-25

1. Introduction

With the Android platform's continuous development and the application field's expansion, the threat of malware will also increase daily. Therefore, the research and application prospects of Android malware detection are still very broad. A homogenous graph means that the relationships between all nodes in the graph are the same; that is, the connection mode and properties are the same between nodes. The study of homogeneous graphs focuses on similarities and commonness between nodes and is often used in social network analysis, bioinformatics, etc. [1]. Node aggregation is the merging of a set of nodes in a homogeneous graph into one hypernode to reduce the size and complexity of the graph. And GraphSAGE (Graph Sample and Aggregating) is a representation learning framework used for graph data. The current algorithm has some limitations in the node aggregation process and fails to fully use semantic relations and feature information between nodes. Therefore, further algorithm improvements are needed to improve the expressive power of node aggregation to meet the processing challenges of large-scale homogeneous graphs. Long and short-term memory (Long short-term memory, LSTM) is widely used in language models, machine translation, text classification, emotion analysis, named entity recognition and other tasks. Its memory unit and gating mechanism enable the LSTM model to capture the long-term dependence of sentence and text sequences effectively [2]. Malware detection systems often face high error rates and false alarm rates. The detection system may misidentify legitimate software as malware or wrongly judge malware software as legitimate software. This can limit users' legal behavior or prevent potential malware from detecting [3]. Therefore, the paper studies the analysis and behaviour modelling of Android malware features and proposes a new Android malware detection method combined with LSTM (Combining mean with LSTM, MLSTM) aggregator, aiming to realize the efficient and accurate detection of Android malware. The study mainly includes four parts. The first part is a review of Android malware detection and LSTM-related research. The second part is the Android malware detection model based on MLSTM. The first section is the data pre-processing and feature extraction of Android malware, and the second section is the malware detection model based on MLSTM. The third part is the result analysis of the malware detection model based on MLSTM, the first section is the performance analysis of the MLSTM model, and the second section is the malware detection results of MLSTM. The fourth part is divided into the conclusion of the Android malware detection model based on MLSTM.

2. Related Works

In recent years, malware has been an increasing threat to mobile devices and the growing number of users on Android platforms. Therefore, there is an urgent need to find efficient and reliable detection methods, and many scholars have conducted relevant research. Yuan et al. developed a lightweight Android malware detector for the smooth training of mobile devices to avoid communication overhead and privacy leakage and improve detection accuracy and robustness. The results show that the detector is practical [4]. To detect the malware of the Android application, Mahindru and Sangal built the Android malware detection framework through

machine learning technology to protect users' privacy and the system's integrity. They used the features in the feature selection method to complete the training. The results show that the detection rate of the framework is 98.8% [5]. The team of Zhang designed an Android malware detection method combined with a hybrid sequence to detect all potential code execution paths of the system, called the dynamic system and static operating code, and processed the sequence through natural language. The results show the method has flexibility and effectiveness [6]. To reduce the dependence of feature learning on prior knowledge, Zhu and other researchers built a deep-learning malicious software detection framework combined with the denoising autoencoder to learn rich features and improve detection performance. The results show that the highest accuracy of this framework is 94.46% [7]. Mahindru and Sangal conceived a framework for Android malware detection combined with unsupervised machine learning to improve detection efficiency. They used different feature sorting methods to select feature measurement performance parameters and implement a self-organization mapping algorithm. The results show that the detection rate of this framework reached 98.7% [8]. To avoid malware threats to device security, professionals such as Zhu et al. implement principal component analysis of feature subsets through the stack integration framework of bootstrapping sample technology and learn the implied supplementary information, which shows that the average accuracy of this method is 94.92% [9].

Many researchers have studied LSTM in depth and proposed various improvement methods to improve their prediction performance and applicability to function in more fields. To predict the individual bidding of competitors in the power market, Guo's team built a set supply curve (ASC) prediction framework based on the LSTM model, which simplified the high dimension of ASC and fixed the unstructured data format. The results showed that the framework has good prediction performance [10]. Kaselimi and other scholars designed an adaptive bidirectional LSTM model through a modular mode to identify a given total power signal equipment and select and drive the best configuration to avoid the multi-dimensional problem of increasing the number of devices. The results show that this method has some practical [11]. To carry out the charging building energy management, Zhou and other researchers designed a recurrent neural network and LSTM system to carry out additional output filtering processes so as to improve decision performance and reduce prediction and computational pressure. The results show that the system is better practical [12]. For the effective processing of resistance random access memory, professionals use LSTM to process tiles to improve efficiency to reduce the requirements of analog-to-digital converter and hardware constraints and enhance LSTM pruning so that the error affects the inference accuracy. The results show that this method is robust [13]. To identify group activity, Tang et al. developed an LSTM model with global and spatiotemporal context consistency constraints to capture correlated movements, suppress the updating of unrelated movements and control memory states. The results show that the method achieves a certain effectiveness [14]. For the situational awareness level prediction of the power system, Wang and other scholars built a convolutional neural network CNN-LSTM model for synchronous learning of temporal and spatial characteristics. They completed the collaborative data mining of spatiotemporal measurement data to avoid the unstable operation state of the power system. The results show that the prediction accuracy of this model is high [15].

In conclusion, the current Android malware detection methods and LSTM have made some progress in improving the detection efficiency and accuracy. However, the information loss in the GraphSAGE framework has not been improved. Therefore, to realize the efficient and accurate detection of Android malware, the analysis and behavior modelling of Android malware features are studied, and an Android malware detection method is proposed to combine mean aggregator and LSTM.

3. Android Malware Detection Model Based on MLSTM

We study the pre-processing and feature extraction of Android malware data, and we build the Android malware detection model of MLSTM by two-order neighbourhood aggregation.

3.1 Data pre-processing and feature extraction of Android malware

Android malware refers to malicious programs or applications aimed at the Android operating system, aiming to cause damage or illegally gain benefits to users' devices, data and privacy. It is generally used for viruses, Trojan horses, adware, spyware, and other forms. Android malware requests excessive permissions to get users' sensitive information or perform malicious actions. These permissions may include accessing contacts, making calls, obtaining location information, and others [16]. However, Android malware can cost a lot of system resources, such as CPU, memory, and battery, and abnormal behavior can be detected by monitoring resource usage. Therefore, the data characteristics of Android malware samples involve permission requests, behavior mode, code characteristics, network communication resource occupation, etc. Studying these characteristics can provide a basis for constructing effective malware detection methods. The standard feature extraction methods in malware detection include static and dynamic [17]. To complete data processing outside the image, GraphSAGE is used to gradually propagate and integrate the feature information of the nodes to obtain a more prosperous and higher-level representation. GraphSAGE Model schematically, as shown in Fig. 1.

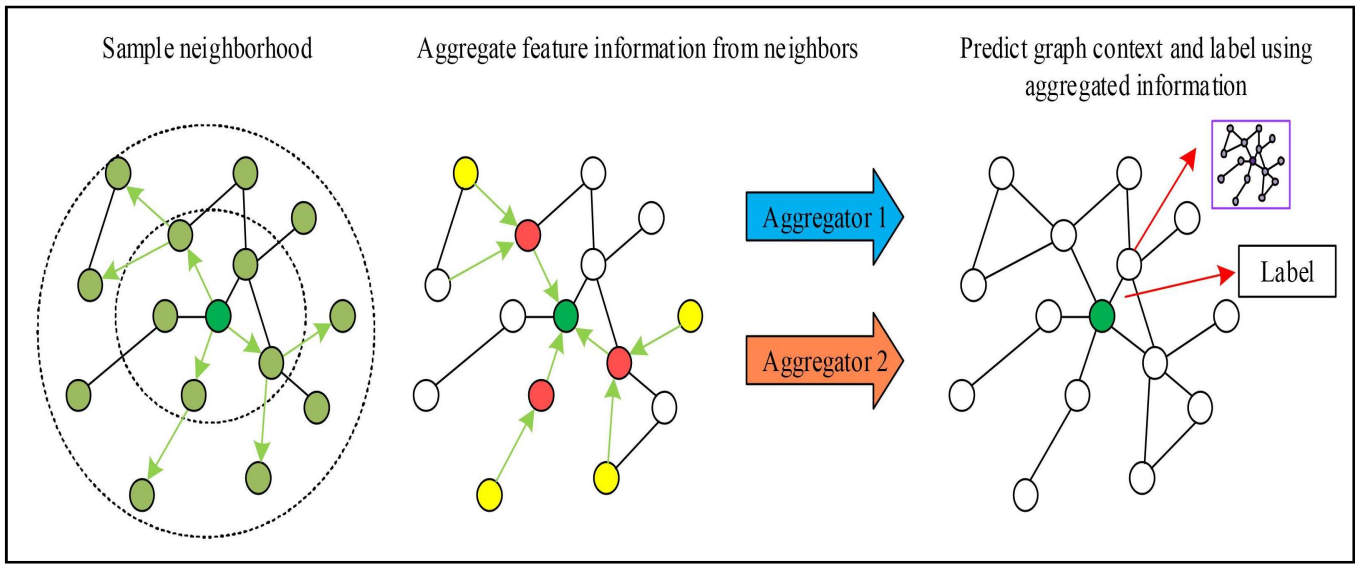


Figure 1. GraphSAGE model diagram

In Figure. 1, GraphSAGE learns a low-dimensional representation of nodes by sampling and aggregating features of neighbor nodes to perform tasks on the graph, such as node classification, link prediction, etc. The core idea is to enrich the representation of the target nodes by sampling the features of the neighbor nodes and aggregating these features onto the target node. It aggregates based on the features of the neighbors' nodes and uses these aggregated features to update the representation of the target nodes. In t+1 times, the neighbors of the target node i features converge $m_i^{(t+1)}$, as shown in Eq. (1).

$$m_i^{(t+1)} = \sum_{j \in N(i)} M_k\left(h_i^{(k-1)}, h_j^{(k-1)}, e_{ij}\right) \quad (1)$$

In Eq. (1), $j \in N(i)$ is the neighbor of the target node i. In $t+1$ times, the neighbor feature update process $h_i^{(t+1)}$ of the target node is shown in Eq. (2).

$$h_i^{(t+1)} = U_t\left(h_i^{(t)}, m_i^{(t+1)}\right) \quad (2)$$

The model of the node aggregation algorithm can capture a higher level of semantic information through the neighbourhood information between the nodes to improve the model's robustness. However, in the process of node aggregation, the current algorithm cannot fully use the semantic relations and feature information between nodes. Therefore, homogeneous maps of third-party library calling features and authority features are constructed to improve the expression ability of node aggregation. Based on the graph convolutional neural network, each Android software node is learned and combined with the mean aggregator of LSTM. Then, the feature vectors are classified in the classifier. The application feature interface (application program interface, API) in Pscout and 5000 [18] were selected using the chi-square test. API calling is the process by which developers interact and communicate with other software, libraries, or services using the interfaces and methods provided by the API.

The specific functions, obtaining the required data, and realizing the data exchange between different systems can be realized through the API calling. The flow of feature extraction is schematic, as shown in Fig. 2.

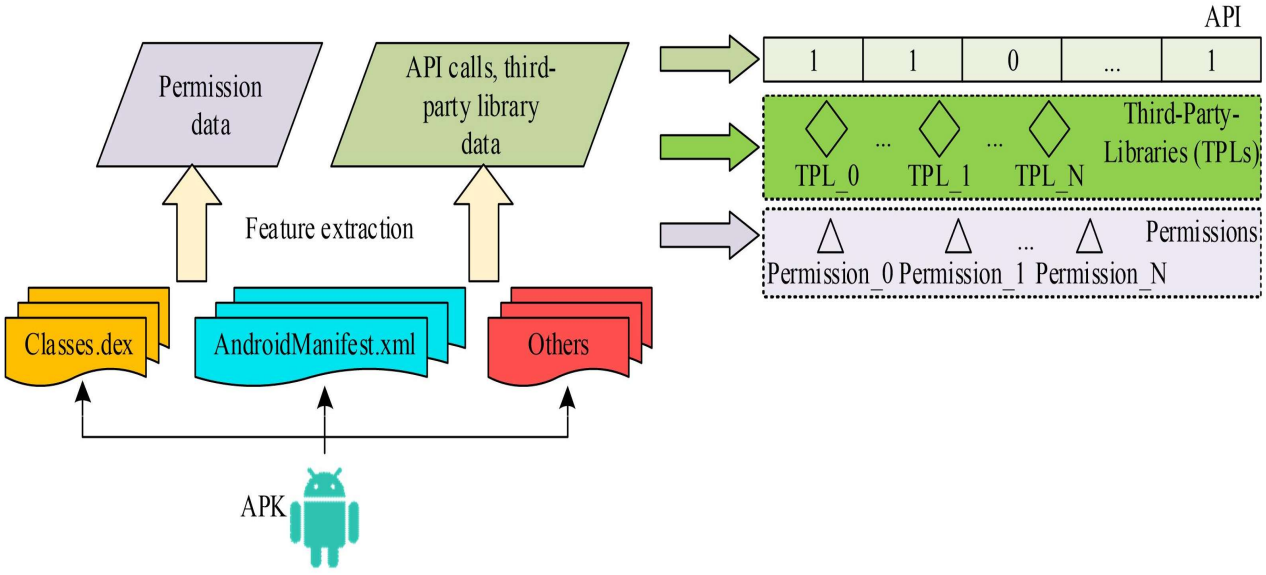


Figure 2. Schematic diagram of the feature extraction process

In Fig. 2, the Android software internal file Classes. The *dex* can complete API calls and third-party library data, such as AndroidManifest—complete XML for permission feature extraction. The original embedding of software nodes in the homogeneous graph is API characteristics, and third-party library characteristics and permission characteristics complete the establishment of connections between software nodes.

3.2 Malware detection model based on MLSTM

In GraphSAGE, the goal of the aggregator is to aggregate the features of the neighbor nodes of the target node and generate a new node representation. Mean aggregators are simple and efficient, with slight information loss. Where the vector x_v^k expression of mean aggregation updates, as shown in Eq. (3).

$$x_v^k \leftarrow \sigma\left(W \cdot MEAN\left(\left\{x_v^{k-1}\right\} \cup \left\{x_u^{k-1}, \forall u \in S(v)\right\}\right)\right) \tag{3}$$

In Eq. (3), the input sample is x, σ is the Sigmoid activation function, x_u^{k-1} is for the current vector, x_v^{k-1} is the sampled neighbor node information, W is the weight vector. However, the mean aggregation cannot handle the long-distance dependence very well. LSTM aggregation can capture the timing information of the nodes, and the working principle is to use the LSTM model to model the feature sequence of the neighbor nodes to obtain a new representation of the target nodes. Therefore, to improve the performance of GraphSAGE on tasks such as node classification and link prediction, the study of a MLSTM aggregator was designed. The neighbor feature matrix completes the averaging processing, and after the hidden layer embedding vector changes, it enters the LSTM layer and finally outputs the k-1 layer vector. The LSTM structure is schematized, as shown in Figure 3.

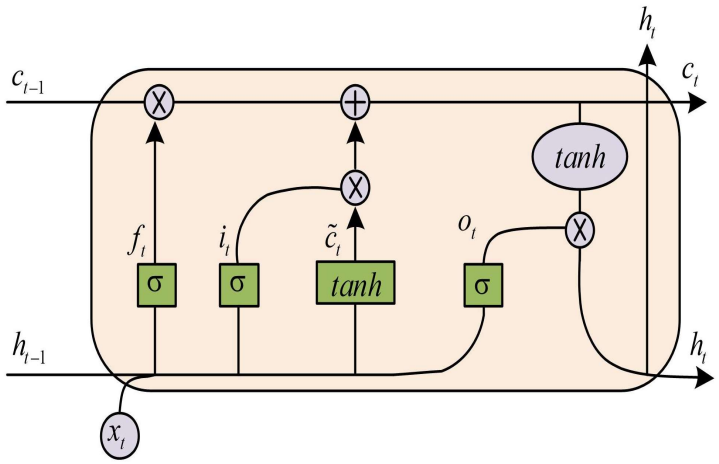


Figure 3. LSTM structural diagram

In Fig. 3, LSTM has three gating units: input gate, forgetting gate and output gate. The input gate determines which information in the input of the current moment needs to be added to the memory state by taking the hidden state h_{t-1} of the previous moment and the input x_t of the current moment as input. After a Sigmoid function and a Tanh function, respectively, we output a value between 0 and 1 and a value between-1 and 1, which represent the proportion and information to be added, respectively. Its output i_t is shown in Eq. (4).

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \tag{4}$$

In Eq. (4), b_i is the bias of the input gate, and the weight matrix of the input gate is W_i. The Tanh function calculates the new candidate vector $\tilde{c}_t$, as shown in Eq. (5).

$$\tilde{c}_t = \mathrm{Tanh}\left(W_c\left[h_{t-1}, x_t\right] + b_c\right) \quad (5)$$

In Eq. (5), b_c is $\tilde{c}_t$ update. The bias, for W_c update. The weight of the matrix. The forgetting gate determines which information needs to be forgotten in the memory state of the previous moment and its input. h_{t-1} and x_t After a Sigmoid function, a value between 0 and 1 is output, which represents the proportion of information that needs to be retained or forgotten. The formula f_t is as shown in Eq. (6).

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) \quad (6)$$

In Eq. (6), b_f is the bias of the forgetting gate, and the weight matrix of the forgetting gate is W_f .The memory element to achieve the state update, updated after the result c_t, As shown in Eq. (7).

$$c_t = f_c \cdot c_{t-1} + i_t \cdot \tilde{c}_t \quad (7)$$

The value o_t of the output gate in the memory element state is shown in Eq. (8).

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \quad (8)$$

The final output h_t, as shown in Eq. (9).

$$h_t = o_{t\cdot} \cdot \mathrm{Tanh}\left(c_t\right) \quad (9)$$

In Eq. (9), the bias of the output gate is b_o, W_o is the weight matrix of the output gate. In graph convolutional neural networks, two-order neighborhood aggregation can capture richer context information, stronger expression ability and context perception, improving the model's performance and robustness [19]. Therefore, MLSTM implements a two-order neighborhood aggregation process, as shown in Fig. 4. *K* In Fig. 4, 3D convolution operation, two-order neighborhood aggregation, first to the sampling of the first neighbor characteristics averaging, after matrix multiplication with the hidden layer state, and dimension transformation by LSTM output vector, the original embedding in the second dimension, the vector through the Relu activation function as the hidden vector of the next layer, the final target node embedding vector by aggregation again. Later, the embedding of the central node is stitched with the aggregated neighbor node embedding by updating the function, and the Relu activation function and Dropout operation are adopted to return the updated vector finally. Finally, the embedding vector learned by the nodes is sent into the classifier to complete the classification. The MLSTM malware detection model for the study design is shown in Fig. 5.

In Figure. 5, the original feature selection software of Android software selects the single feature connection between the software to build the homogeneous graph. Under the GraphSAGE framework, the neighbor node information of the target node is extracted, and the information aggregation is realized after utilizing the improved aggregator. For the output result splicing, the processing is completed in the Relu activation function and the Dropout layer, and then the feature vectors are classified in the classifier, including malware and benign software. The Fast Gradient Sign Method (FGSM) is an attack method that generates counter samples for the malware detection model. The principle is to use the gradient information to find the direction of the model prediction result in the input sample and perturb in this direction so that the model is misjudged. The FGSM formula is expressed, as shown in Eq. (10).

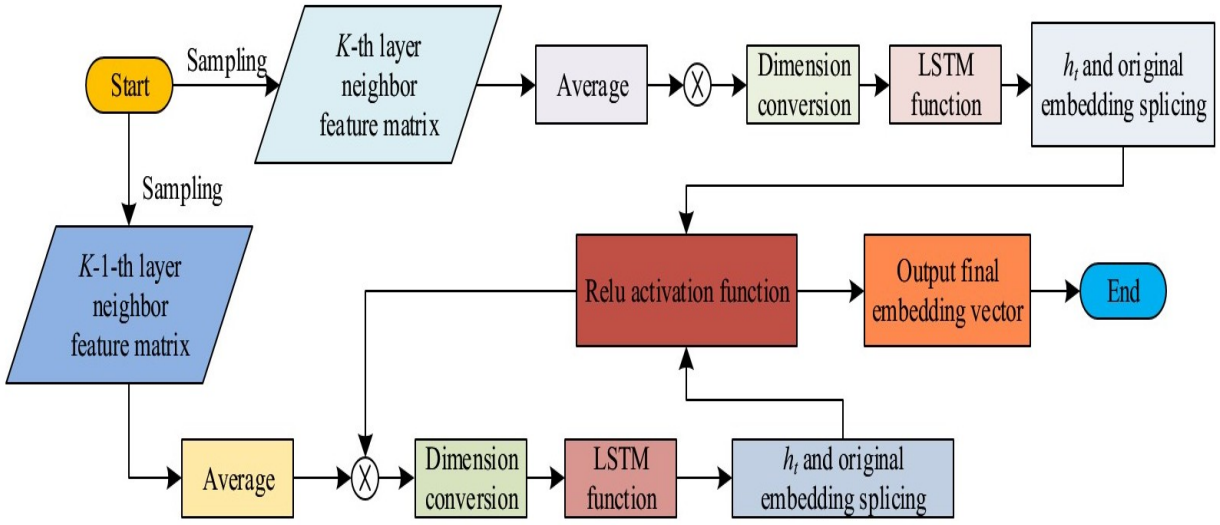


Figure 4. Schematic diagram of MLSTM implementing second-order neighborhood aggregation process

$$\hat{x} = x + \in sign\left(\nabla_x J\left(f\left(x\right), y\right)\right) \tag{10}$$

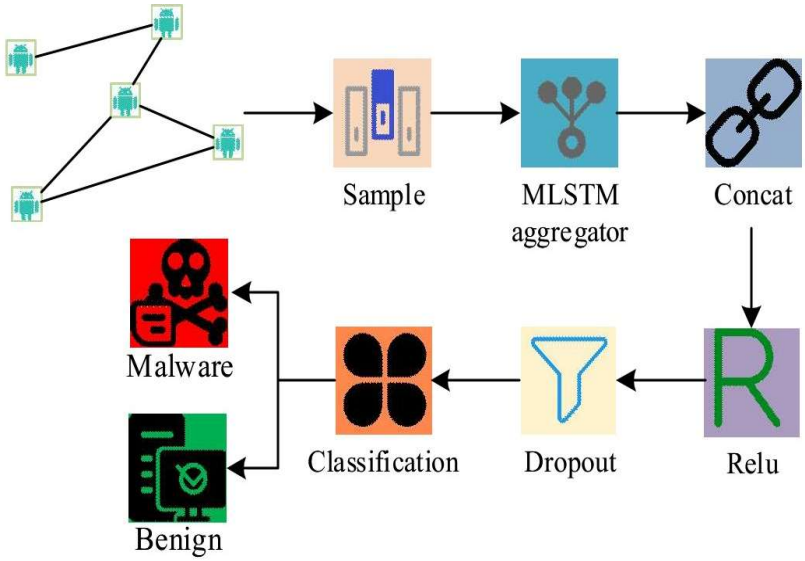


Figure 5. MLSTM malicious software detection model

In Eq. (10), the size of the disturbance is $\in \cdot J\left(f\left(x\right), y\right)$ is the loss function.∇_x is calculate the gradient operation and find the maximum gradient *d*, *d* =10, *d* = 20 Eq. (11) shows the mean absolute error (MAE) in the model detection evaluation index.

$$MAE = \sum \left|P - Q\right| / n \tag{11}$$

In formula (11), P is the predicted value, Q is the actual value, and the number of samples is n. Root mean squared error (RMSE), as shown in Eq. (12).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \tag{12}$$

In Eq. (12), y_i and $\hat{y}_i$, The predicted and actual values, respectively.

4. Performance Analysis of the Malware Detection Method Based on MLSTM

The influence of different parameters on the performance of the MLSTM model was analyzed, and the detection accuracy with different aggregators was compared to verify the robustness of the improved model's detection results.

4.1 MLSTM model performance analysis

The experimental running platform selects Windows Server 2016 Datacenter, the version is Python 3.7, the code editor uses PyCharm 2019.3.5, the open source Python machine learning library PyTorch building model, and the optimizer selects Adam. Experimental environment configuration information is shown in Table 1.

Hardware and software	Illustrate
CPU	Intel (R) Xeon (R) CPU E5-2678 v3 @ 2.50 GHz
Running Platform	Windows Server 2016 Datacenter
Hard Disk	100 G
Memory	8 G
Deep Learning Framework	PyTorch
Programming Language	Python 3.7, PyCharm 2019.3.5

Table 1. Experimental Environment Configuration Information

The dataset used in the experiment included 50.17% benign samples and 49.835 malicious samples, with a partition ratio of 8: 2. The MLSTM model was compared with four modes: mean aggregator (Mean aggregator, MA), LSTM aggregator (LSTMA), maximum pooling aggregator (Maximum pooling aggregator, Max-PA), and mean pooling aggregator (Mean pooling aggregator, Middle-PA). When studying the different models, only the aggregators are inconsistent, and the other parameters remain unchanged. The comparison of recall, precision and F1 scores of various models under authority characteristics and third-party library features are shown in Figure 6.

In Figure. 6(a), the recall, precision, and F1 scores of the MLSTM model are 0.9924, 0.9964, and 0.9937, respectively. Since the Max-PA model can screen out the most essential features in the neighbor nodes and is not dis turbed by other features, it has the highest accuracy of 0.9969. However, the MLSTM model, averaging

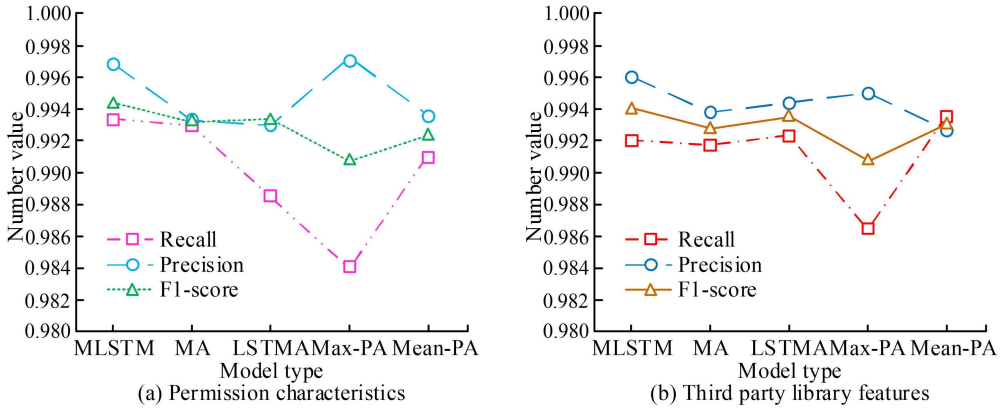


Figure 6. Comparison of recall, precision, and F1-score results of different models under permission features and third-party library features

neighbor node features, may ignore some important information and be limited by sequence length and memory ability. In Fig. 6(b), the recall, precision, and F1 score of the MLSTM model are 0.9909, 09.9975, and 0.9942, respectively. The Mean-PA model has the highest recall of 0.9934 because its aggregator combines the mean of neighbor node features with the maximum, considering the overall and important features of neighbor nodes. At this time, the Max-PA model is under the contrast permission feature. The accuracy is reduced by 0.9959. Therefore, the MLSTM model showed better generalization performance. To evaluate the robustness of the model, the MLSTM model and MA, LSTMA, Max-PA, and Mean-PA models were compared in unknown samples when the results of different models under permission characteristics and third-party library features are shown in Fig. 7. d = 10, d = 20.

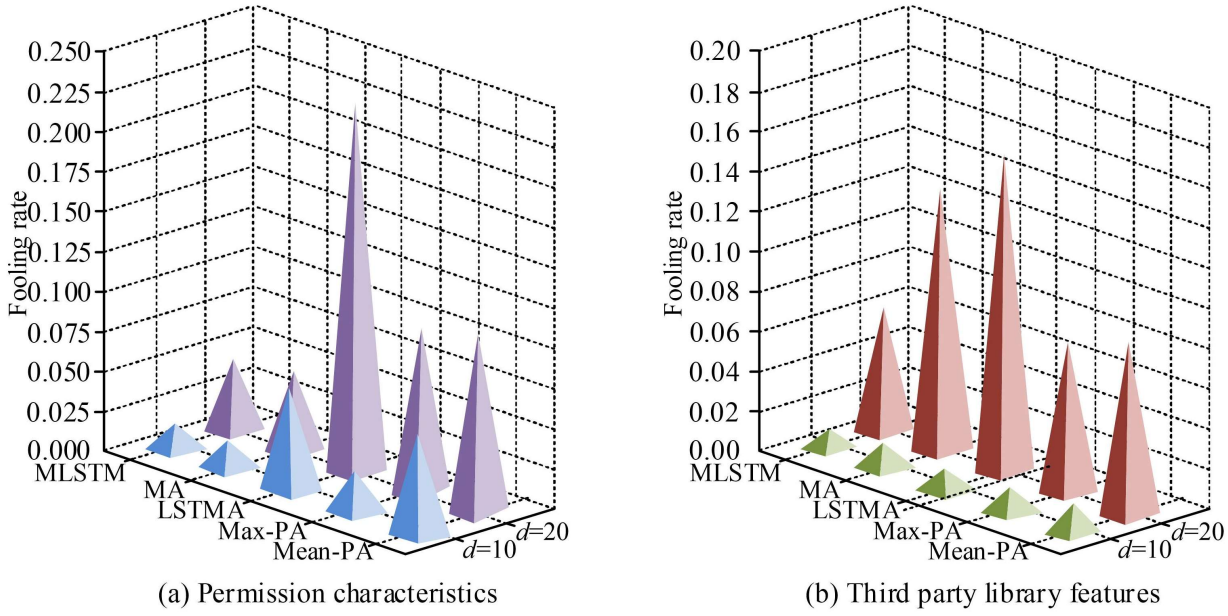


Figure 7. The fooling rate results of different models with different d values under permission features and third-party library features

In Figure. 7(a), under the permission characteristics, the LSTMA model at d =10 and d =20 has the largest fool rate, respectively 0.0684 and 0.2478. At and , the MLSTM model has the lowest fool rate compared to the other four models, respectively, 0.0178 and 0.0506. At and , the fooling rate of the LSTMA model is approximately 3.85 times and 4.90 times that of the MLSTM model, respectively. In Fig. 7(b), under the characteristics of the third-party library, the MLSTM model is 0.0097 and 0.0640, respectively. When The LSTMA model has the highest fool rate, 0.1590. To verify the effect of the number of neighbor samples on the detection accuracy of the MLSTM model, the number of neighbor samples was compared from 50 to 100. The Max-PA model and Detection accuracy of the MLSTM model are shown in Fig. 8.

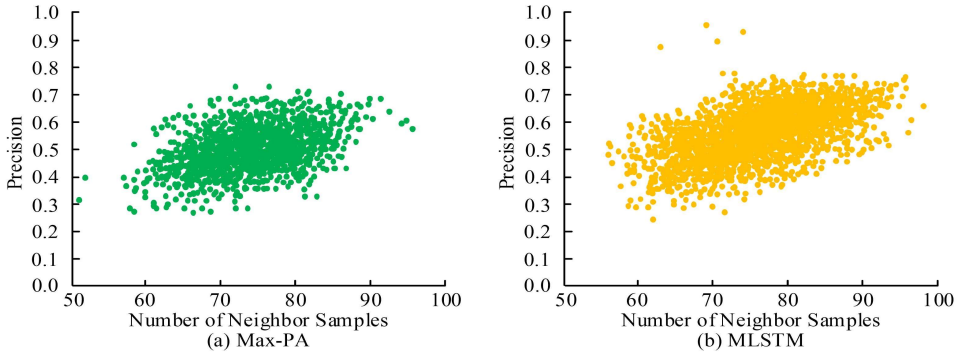


Figure 8. Comparison of detection accuracy between Max-PA model and MLSTM model

In Figure 8(a), the Max-PA model MAE is 1.82 and RMSE is 2.26, and when the sampling number is 75, the average accuracy is only 0.47. The model accuracy improved as the number of neighbor samples increased. In Fig. 8(b), the MAE of the MLSTM model is 1.64, and the RMSE is 2.08, smaller than the Max-PA model, indicating a more minor error and better detection accuracy. The more sample data, the better the model fit and the higher the accuracy. Considering that the composition similarity and the number of sampling layers influence the model detection results, the study designed the Euclidean distance similarity, Max-PA model and cosine similarity. The MLSTM model was presented under permission and third-party library characteristics. The prediction results are shown in Table 1.

Characteristic	Model	Composition similarity		Number of sampling layers		
		Euclidean distance similarity	Cosine similarity	1	2	3
Permission characteristics	Max-PA	0.52	0.50	0.33	0.52	0.41
	MLSTM	0.56	0.53	0.30	0.56	0.35
Third party library features	Max-PA	0.55	0.53	0.35	0.55	0.44
	MLSTM	0.59	0.56	0.36	0.59	0.42

Table 2. Comparison of detection results for different composition similarity and sampling layers between the Max-PA model and MLSTM model under different Characteristics

In Table 2, Euclidean distance similarity detection was higher in Authority and third-party library features under the Max-PA and MLSTM models. The reason is that the Euclidean distance similarity can effectively measure the similarity between features. Before the Euclidean distance, the features are standardized to eliminate the dimensional difference between feature values. The number of sampled layers determines how many layers of neighbor nodes need to be considered during the aggregation process of each node. Max-PA model sum under the third-party library features. The MLSTM model showed the highest detection results of 0.55 and 0.59, respectively. Increasing the number of sampled layers can provide broader and deeper neighbor node information, contributing to the richness and accuracy of the node representation. This can enable the model to capture better the context information and relationship of the malware nodes in the graph, thus improving the detection ability and classification accuracy of the malware. However, too many sampled layers will increase the complexity of computation and storage, which may lead to increased training time.

4.2 Malware detection results of MLSTM

Fig. 9 shows the detection time for the multiple quantities of the original system and the improved system, as well as the improved system's time saving and improvement rate, to determine the application performance of the designed MLSTM system.

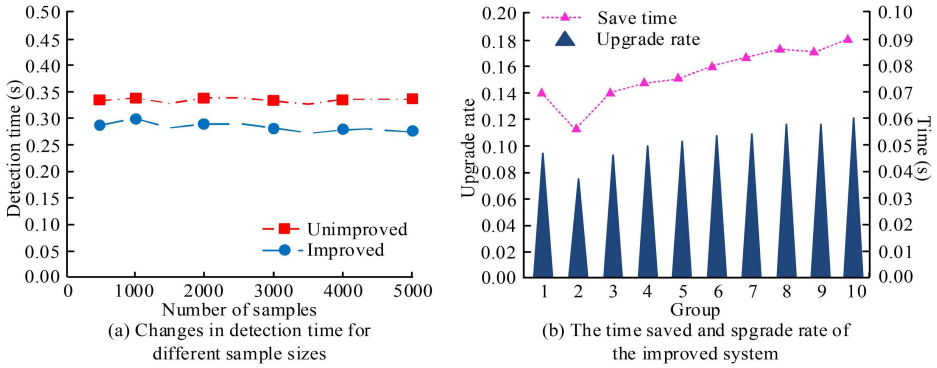
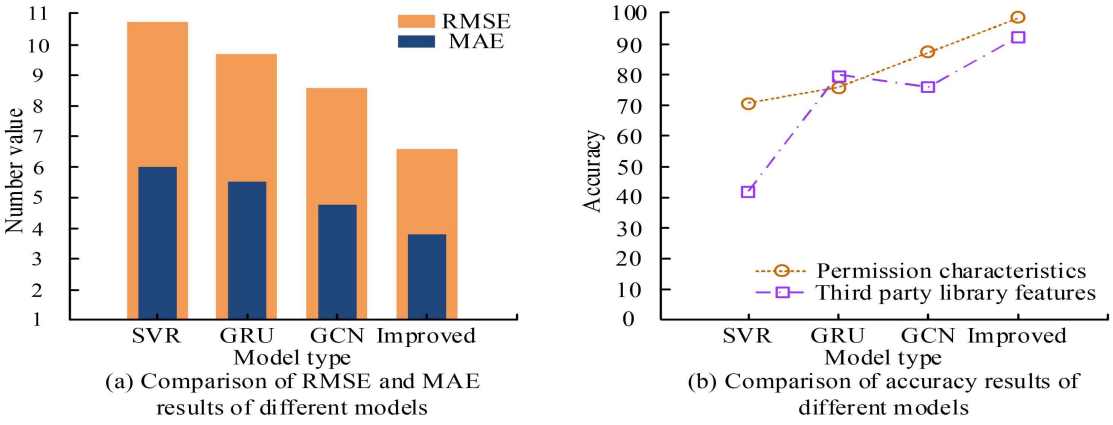


Figure 9. Comparison of running time and upgrade rate between improved and unimproved systems

In Figure. 9(a), with the increase of the number of samples, the detection time of the system is always in a relatively stable state, and the average detection time of the improved and unimproved systems is respectively 0.274 s, 0.336 s. In contrast, the improved detection efficiency of the enhanced system is more prominent. In Fig. 9(b), the average time saving of the improvement system is about 0.05 s, regardless of the number of samples. Overall, the improvement rate of malware detection by the improved system increases with the number of samples, with a rate of up to 18.2%. Group 2 has a low improvement rate, which may be because the input data takes a long time in preprocessing. To compare the detection performance of the improved model, it was compared with support vector regression (Support Vector Regression, SVR), gating cycle unit (Gated Recurrent Unit, GRU), graph convolutional network (Graph convolution Network, GCN), MAE and RMSE results of different models, as well as the accuracy under permission characteristics and third-party library features, as shown in Fig. 10.

In Fig. 10(a), the MAE and RMSE of the improved model are 3.84 and 6.26, respectively. In contrast, SVR has the highest MAE and RMSE, indicating the worst and best detection performance of the improved model. In Fig. 10(b), the accuracy of improved models under permission and third-party library features is higher than other models, with 98.89% and 92.65%, respectively. The accuracy of the GRU permission and third-party library

features is 76.08% and 79.45%, respectively. The reason may be that the data quality of the permission feature is poor, including noise or incomplete, which may cause the model to be unable to learn the effective mode, thus reducing the accuracy accurately.



(a) Comparison of RMSE and MAE results of different models

(b) Comparison of accuracy results of different models

Figure 10. MAE and RMSE results and accuracy of different models

5. Conclusion

To improve the efficiency of Android malware detection, an Android malware detection method of MLSTM is conceived and used to enhance the model robustness by two-order neighborhood aggregation. The results show that the recall, precision, and F1 scores under the MLSTM model permission feature are 0.9924, 0.9964, and 0.9937, respectively. The Max-PA model had the highest accuracy of 0.9969. The recall, precision, and F1 scores under the third-party library features of the MLSTM model are 0.9909, 0.9975, and 0.9942, respectively. The recall of the Mean-PA model was 0.9934, and the Max-PA model precision was 0.9959. In contrast, the MLSTM model showed better generalization performance. The LSTMA model at $d=10$ and $d=20$ has the highest fool rate, with 0.0684 and 0.2478, respectively. The MLSTM model has the lowest fool rate of the other four models, when and with 0.0178 and 0.0506, respectively. Euclidean distance similarity detection results between the Max-PA and MLSTM models were higher under permission and third-party library features. The number of sampled layers determines how many layers of neighbor nodes need to be considered during the aggregation process of each node. When the number of sampling layers is 2, the Max-PA and MLSTM models under the third-party library features are 0.55 and 0.59, respectively. The average accuracy of the Max-PA model was only 0.47. The model accuracy improved as the number of neighbor samples increased. The MLSTM model has a small MAE of 1.64 and an RMSE of 2.08, indicating a better detection accuracy. The more sample data, the better the model fit and the higher the accuracy. However, the study time was short without considering the environmental impact, which was improved and extended in further studies.

References

[1] Dhalaria, M., Gandotra, E. (2021). Android malware detection techniques: A literature review. *Recent Patents on Engineering, 15*(2), 225–245.

[2] Kouliaridis, V., Barmpatsalou, K., Kambourakis, G., Chen, S. (2020). A survey on mobile malware detection techniques. *IEICE Transactions on Information and Systems, 103*(2), 204–211.

[3] Xu, D., Peng, H., Wei, C., Shang, X., Li, H. (2021). Traffic state data imputation: An efficient generating method based on the graph aggregator. *IEEE Transactions on Intelligent Transportation Systems, 23*(8), 13084–13093.

[4] Yuan, W., Jiang, Y., Li, H., Cai, M. (2019). A lightweight on-device detection method for Android malware. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51*(9), 5600–5611.

[5] Mahindru, A., Sangal, A. L. (2021). MLDroid – Framework for Android malware detection using machine learning techniques. *Neural Computing and Applications, 33*(10), 5183–5240.

[6] Zhang, N., Xue, J., Ma, Y., Zhang, R., Liang, T., Tan, Y. A. (2021). Hybrid sequence-based Android malware detection using natural language processing. *International Journal of Intelligent Systems, 36*(10), 5770–5784.

[7] Zhu, H. J., Wang, L. M., Zhong, S., Li, Y., & Sheng, V. S. (2021). A hybrid deep network framework for Android malware detection. *IEEE Transactions on Knowledge and Data Engineering, 34*(12), 5558–5570.

[8] Mahindru, A., Sangal, A. L. (2022). SOMDROID: Android malware detection by artificial neural network trained using unsupervised learning. *Evolutionary Intelligence, 15*(1), 407–437.

[9] Zhu, H., Li, Y., Li, R., Li, J., You, Z., Song, H. (2020). SEDMDroid: An enhanced stacking ensemble framework for Android malware detection. *IEEE Transactions on Network Science and Engineering, 8*(2), 984–994.

[10] Guo, H., Chen, Q., Zheng, K., Xia, Q., Kang, C. (2021). Forecast aggregated supply curves in power markets based on LSTM model. *IEEE Transactions on Power Systems, 36*(6), 5767–5779.

[11] Kaselimi, M., Doulamis, N., Voulodimos, A., Protopapadakis, E., Doulamis, A. (2020). Context-aware energy disaggregation using adaptive bidirectional LSTM models. *IEEE Transactions on Smart Grid, 11*(4), 3054–3067.

[12] Zhou, H., Zhou, Y., Hu, J., Yang, G., Xie, D., Xue, Y., Nordström, L. (2021). LSTM-based energy management for electric vehicle charging in commercial-building prosumers. *Journal of Modern Power Systems and Clean Energy, 9*(5), 1205–1216.

[13] Han, J., Liu, H., Wang, M., Li, Z., & Zhang, Y. (2019). ERA-LSTM: An efficient ReRAM-based architecture for long short-term memory. *IEEE Transactions on Parallel and Distributed Systems, 31*(6), 1328–1342.

[14] Tang, J., Shu, X., Yan, R., Zhang, L. (2019). Coherence constrained graph LSTM for group activity recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(2), 636–647.

[15] Wang, Q., Bu, S., He, Z., Dong, Z. Y. (2020). Toward the prediction level of situation awareness for electric power systems using CNN-LSTM network. *IEEE Transactions on Industrial Informatics, 17*(10), 6951–6961.

[16] Nsugbe, E. (2023). Toward a self-supervised architecture for semen quality prediction using environ-

mental and lifestyle factors. *Artificial Intelligence and Applications, 1*(1), 35–42.

[17] Rana, M. S., Sung, A. H. (2020). Evaluation of advanced ensemble learning techniques for Android malware detection. *Vietnam Journal of Computer Science, 7*(2), 145–159.

[18] Zhang, Z., Li, Y., Dong, H., Gao, H., Jin, Y., Wang, W. (2020). Spectral-based directed graph network for malware detection. *IEEE Transactions on Network Science and Engineering, 8*(2), 957–970.

[19] Mokayed, H., Quan, T. Z., Alkhaled, L., Sivakumar, V. (2023). Real-time human detection and counting system using deep learning computer vision techniques. *Artificial Intelligence and Applications, 1*(4), 221–229.