BILINGUAL PUBLISHING GROUP

Journal of Computer Science Research

https://journals.bilpubgroup.com/index.php/jcsr

ARTICLE

A Multi-model Fusion Strategy for Android Malware Detection Based on Machine Learning Algorithms

Shuguang Xiong [1]***, Huitao Zhang*** [2*]

[1] *Microsoft Inc., Way,Redmond,Washington 98052-6399,United States Of America*

[2] *Northen Arizona University, San Francisco St, Flagstaff, AZ 86011, United States Of America*

ABSTRACT

In the digital age, the widespread use of Android devices has led to a surge in security threats, especially malware. Android, as the most popular mobile operating system, is a primary target for malicious actors. Conventional antivirus solutions often fall short in identifying new, modified, or zero-day attacks. To address this, researchers have explored various approaches for Android malware detection, including static and dynamic analysis, as well as machine learning (ML) techniques. However, traditional single-model ML approaches have limitations in generalizing across diverse malware behaviors. To overcome this, a multi-model fusion approach is proposed in this paper. The approach integrates multiple machine learning models, including logistic regression, decision tree, and K-nearest neighbors, to improve detection accuracy. Experimental results demonstrate that the fusion method outperforms individual models, offering a more balanced and robust approach to Android malware detection. This methodology showcases the potential of ensemble techniques in enhancing prediction accuracy, providing valuable insights for future research in cybersecurity.

Keywords: Component; Multi-model fusion; Malware detection; Machine learning

1. Introduction

In the digital age, the widespread use of Android devices has led to a surge in security threats, especially malware. As the most popular mobile operating system worldwide, Android is a primary target for malicious actors. The emergence of sophisticated malware strains capable of bypassing conventional

*CORRESPONDING AUTHOR:
Huitao Zhang, Northen Arizona University, San Francisco St, Flagstaff, AZ 86011, United States Of America; Email: hz345@nau.edu

ARTICLE INFO
Received: 15 May 2024 | Revised: 19 May 2024 | Accepted: 21 May 2024 | Published Online: 31 May 2024
DOI: https://doi.org/10.30564/jcsr.v6i2.6632

CITATION
Xiong, S.G., Zhang, H.T., 2024. A Multi-model Fusion Strategy for Android Malware Detection Based on Machine Learning Algorithms. Journal of Computer Science Research. 6(2): 1–11. DOI: https://doi.org/10.30564/jcsr.v6i2.6632

detection techniques calls for more advanced defensive approaches. The critical need for robust Android malware detection is undeniable. Malware can result in substantial financial losses, privacy violations, and even jeopardize the security of crucial infrastructure. Traditional antivirus solutions typically use signature-based detection, which is efficient against known threats but often falls short in identifying new, modified, or zero-day attacks. Additionally, the evolving and polymorphic characteristics of contemporary malware render signature-based approaches increasingly outdated.

Android malware detection has evolved over the years, with researchers exploring various approaches, from static analysis, which examines the code without execution, to dynamic analysis, which observes the behavior of applications during runtime. Each method has its benefits and limitations. Static analysis can be fast and safe but often fails to detect sophisticated obfuscation techniques or runtime behaviors. Dynamic analysis is more effective against such techniques but is resource-intensive and can be circumvented by malware that detects the analysis environment. Machine learning (ML) has emerged as a potent tool in cybersecurity due to their excellent performance in many domains [1–8], capable of learning from data to detect patterns indicative of malicious behavior shown in **Figure 1**. For example, Arshad et al. [9] compared 20 Android malware detection tools from the perspectives of static and dynamic analysis, supported by 62 references. Rashidi et al. [10] offered an overview of Android security threats and defenses through static and dynamic analysis, citing 141 references. An extensive review of 124 methods using static analysis of Android apps was conducted in [11], which included 188 references. Furthermore, Tam et al. [12] presented 36 Android malware detection tools, expanding into static, dynamic, and hybrid analysis, with 181 references. Despite these comprehensive surveys of software engineering approaches, the internal ML or deep learning (DL) algorithms were only superficially analyzed. A few surveys have explored traditional ML-based methods, such as in [13,14], where Faruki et al. [15] provided an overview of Android malware penetration and defense strategies, including ML-based detection tools with 117 references. Ebtesam et al. [16] briefly reviewed four ML algorithm families—SVM, NB, perceptron, and Deep Neural Network (DNN)—for Android malware detection. Traditional ML algorithms like K-means, Support Vector Machine, Decision Tree, fuzzy logic, Artificial Neural Network, Gaussian methods, and Meta-heuristic approaches were detailed in [17], focusing on signatures and behaviors.

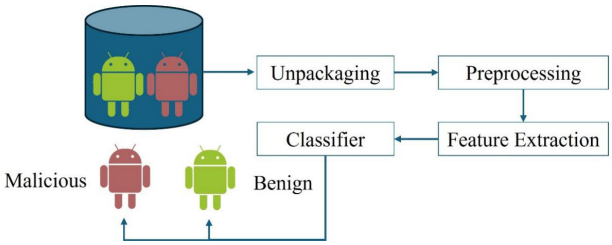

Figure 1. The workflow of machine learning-based Android malware detection.

However, traditional Android malware detection has relied on single machine learning models, which presents several disadvantages. These models can be limited in their ability to generalize across diverse malware behaviors, leading to reduced detection accuracy, especially with new or sophisticated threats. A single model may also be prone to overfitting, where it performs well on training data but fails to predict accurately on unseen data. To address these issues, considering the fusion of multiple models offers significant advantages. Combining different models can harness their individual strengths, leading to more robust and accurate detection across a broader range of malware types. This approach can improve the system's resilience to zero-day attacks and polymorphic malware, which often evade single-model systems. Additionally, model fusion can enhance the adaptability of the detection system, allowing it to evolve with emerging threats more effectively.

In this paper, we develop a multi-model fusion approach for predicting Android malware detection. Initially, the study employs various machine learning models, specifically KNN, decision tree, and logistic regression. Each model is trained with malware data

using different parameters to determine the single best predictive model. Subsequently, these three machine learning models are integrated through fusion and compared with the individual best-performing model. The evaluation criteria clearly indicate that the multi-model fusion method achieves higher accuracy and provides better malware prediction. Additionally, this paper utilizes a decision tree feature importance analysis experiment to identify the key factors influencing malware. Compared to traditional single-model algorithms, this multi-model fusion approach using machine learning could offer valuable insights for future predictions.

This paper is structured as follows: section 2 details the related works of Android malware detection. Following that, section 3 provides the workflow of the proposed method in this study. The experimental results and corresponding discussion are provided in section 4. Finally, section 5 provides a comprehensive conclusion of this paper.

2. Literature review

Android malware detection

With the rapid advancement of mobile internet and smart devices, malware targeting the Android platform has evolved into various forms. Traditionally, Android malware is categorized into types such as trojans, backdoors, worms, botnets, spyware, aggressive adware, and ransomware [18]. Felt et al. [19] further distinguished Android malware based on the motivations behind human behaviors, which include seeking novelty and amusement, selling user information, stealing credentials, making premium-rate calls, sending SMS messages, distributing SMS spam, enhancing search engine optimization, and extracting ransom. Zhou and Jiang [20] analyzed Android malware through the lenses of installation, activation, malicious payloads, and permission exploitation. In its reports, Google [21] employs conservative terminology to describe such software, referring to them as potentially harmful applications (PHAs). In the Google Play Store, these PHAs are classified under various categories, including click fraud, SMS fraud, spyware, toll fraud, trojans, hostile downloaders, backdoors, phishing, privilege escalation, and commercial spyware.

Several studies have also explored the extraction of static features from unique perspectives. Protsenko et al. [22] approaches Android malware detection by analyzing software complexity, extracting 144 features that reflect a program's control flow, data flow, and object-oriented design. Meanwhile, Yang et al. [23] introduces a static feature named a modality vector, generated through a three-step process: behavior graph generation, sensitive node extraction, and modality generation. Xu et al. [24] focuses on malware detection from the perspective of inter-component communication (ICC) among Android applications, extracting features related to ICC from components, explicit intents, implicit intents, and intent filters.

Yang et al. [25] detects malware by analyzing textual features within grayscale images. It begins by unzipping the APK file, converting the files classes.dex, AndroidManifest.xml, resources.arsc, and cert.rsa into 8-bit unsigned integer vectors, and then transforming them into grayscale images for further analysis. Martín et al. [26] emphasizes static features of third-party API calls, which are difficult to obfuscate and can enhance detection accuracy.

Machine learning algorithms are also considered due to their strong feature extraction and prediction performance [27–36]. Shen et al. [37] defines a data flow named Complex-Flows, extracts API call sequences using data flow analysis tools like BlueSeal, generates feature sets in the form of n-grams, and employs an SVM algorithm for malware detection. In Gorla et al. [38], natural language processing (NLP) technology and the k-means algorithm are initially used to process each application's description in the market, combined with extracted sensitive API calls, and finally, an SVM algorithm is applied for malware detection. Lastly, Li et al. [39] adopts an incremental SVM algorithm that utilizes prior information from historical samples to avoid retraining all data when new samples are added, thus enhancing detection efficiency.

3. Method

3.1 Dataset preparation

The dataset used in this study consists of 7,845 entries, each described by 14 features. These features include name, which identifies the Android application; tcp_packets, representing the number of TCP packets sent and received; dist_port_tcp, the number of distinct TCP ports; external_ips, the count of unique external IP addresses contacted; vulume_bytes, the total volume of data sent and received in bytes; udp_packets, the number of UDP packets; tcp_urg_packet, the count of TCP packets with URG flags; source_app_packets, the total packets sent by the application; remote_app_packets, the total packets received from the remote application; source_app_bytes, the total bytes sent by the application; remote_app_bytes, the total bytes received from the remote application; source_app_packets.1, a possibly duplicated column of source_app_packets; dns_query_times, the number of DNS queries made by the application; and type, indicating whether the traffic was benign or malicious.

In the preprocessing phase of our study, we transformed the labels in the dataset into numerical format for compatibility with machine learning models. Specifically, we converted the type feature, which originally categorized traffic as either "benign" or "malicious", into binary labels: "0" for benign and "1" for malicious. Additionally, we partitioned the dataset into a training set and a test set, allocating 70% of the data to the training set to train our model, and reserving 30% for the test set to evaluate the model's performance on unseen data. This division ensures a robust assessment of the predictive capabilities of our developed models.

3.2 The introduction of used machine learning models

Logistic regression

Logistic regression is a fundamental statistical method used primarily for binary classification problems, where the objective is to predict a binary outcome from a set of variables. Originating in the field of statistics, logistic regression has found extensive application in machine learning due to its simplicity, interpretability, and efficiency in scenarios where the response variable is categorical—particularly when it is dichotomous (e.g., yes/no, true/false, positive/negative).

The core idea behind logistic regression is to model the probability of the default category (often labeled as '1') of a binary dependent variable as a logistic function of one or more independent variables (predictors). Unlike linear regression that might predict any value from negative to positive infinity, logistic regression outputs probabilities between 0 and 1 by using the logistic (or sigmoid) function. This function is mathematically represented as $\sigma(z) = \frac{1}{1 + e^{-z}}$, where z is a linear combination of the input features weighted by their coefficients $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$. In logistic regression, the coefficients (β) are estimated using the method of maximum likelihood estimation (MLE). The goal of MLE is to find the set of coefficients that maximizes the likelihood of the observed sample, which effectively means choosing the coefficients that make the observed outcomes most probable under the model. This approach adjusts the coefficients to fit the model such that the predicted probability reflects the actual distribution of the outcomes as closely as possible. Once the model is trained, the logistic regression can predict the probability that a new observation belongs to the default category. These probabilities can then be converted to class predictions by setting a threshold, commonly 0.5 in binary classification, where probabilities above this threshold predict the default category, and those below predict the alternative category.

Decision tree

A decision tree is a versatile machine learning model used for both classification and regression tasks. It is one of the most intuitive and easy-to-understand models, as it mirrors human decision-making processes by splitting data into branches to reach conclusions or predictions. This model's structure

resembles an inverted tree, starting with a root node and expanding into branches and leaves representing decisions and outcomes, respectively.

At the heart of a decision tree is a sequence of questions and decisions about the data. Each internal node in the tree represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label (in classification) or a continuous value (in regression). The paths from root to leaf represent classification rules or regression paths. To build a decision tree, algorithms like ID3, C4.5, Classification and Regression Trees (CART), or others iteratively split the dataset into subsets based on different features. The selection of features and the point at which to split them are determined using metrics such as Gini impurity, entropy, or variance reduction. These metrics help to identify the feature and threshold that most effectively split the data to maximize the homogeneity of the subsets.

K-nearest neighbors

K-nearest neighbors is a straightforward and versatile machine learning algorithm used for both classification and regression tasks. It is a non-parametric method, which means it makes no explicit assumptions about the underlying data distribution. This simplicity and flexibility make KNN a popular choice for many practical applications, particularly when the data is well structured and labeled.

The essence of KNN lies in predicting the label of a new data point by examining the 'k' closest labeled data points from the training set. Here, 'k' represents the number of nearest neighbors the algorithm considers in its prediction process. KNN begins by calculating the distance between the test instance and every instance in the training set, using distance metrics such as Euclidean, Manhattan, or Minkowski. After calculating these distances, it sorts all training instances by proximity and selects the 'k' nearest ones. For classification tasks, the most common class among these neighbors is assigned to the test instance, typically through majority voting. In regression tasks, the algorithm predicts a value based on the average or another statistical measure of the dependent variable among these neighbors.

Random forest

Random Forest is a powerful ensemble learning technique used in machine learning for both classification and regression tasks. It builds on the simplicity and effectiveness of decision trees by combining multiple such trees to improve predictive performance and control overfitting. The fundamental principle behind Random Forest is to create a "forest" of decision trees, each trained on different parts of the same dataset, and then aggregate their predictions to produce a more accurate and stable result than any single tree could provide.

The process begins with the creation of multiple decision trees during the training phase. Each tree in a Random Forest is constructed using a random subset of the training data and features, a method known as "bootstrap aggregating" or bagging. This randomness helps in diversifying the individual trees, reducing the correlation between them, and thereby enhancing the overall model's robustness. When making predictions, for classification tasks, the mode of the classes predicted by individual trees is taken (majority voting), and for regression, the average of the outputs is considered.

3.3 Multi-model fusion strategy for Android malware detection

In this paper, we introduce a novel model fusion strategy designed to enhance the robustness and accuracy of predictive modeling. The process begins by feeding the original dataset into three different models for training. Using the optimal model, predictions are generated for both the training and test sets. The predicted values from the three models—logistic regression, decision tree and K-nearest neighbors—are then aggregated to form a new set of features. The essence of this strategy lies in utilizing these aggregated features as inputs to train and fine-tune the random forest model. The experimental framework is implemented using the sklearn library within the Python3 environment. **Figure 2** illustrates the model fusion process. The following steps detail the model fusion approach: (1) To start, import the

dataset and split it into training and test sets, adhering to a 70:30 ratio. Additionally, set the random state parameter to a fixed value to ensure that the data splits are consistent across all three models. (2) Training three regression models: logistic regression, decision tree and K-nearest neighbors. Iteratively adjust these parameters to find the optimal configuration for each model. (3) Building on the achievement of identifying the optimal single model, aggregating the predictions from these three models to create a new set of features. Use these aggregated features as input to train and fine-tune an enhanced Random Forest model. (4) Evaluating the performance of different models using accuracy and confusion matrix metrics.

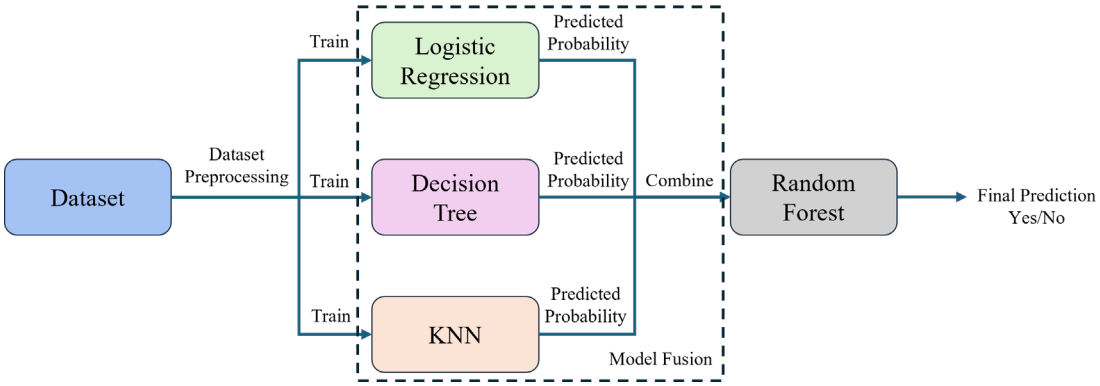

Figure 2. The workflow of the proposed model fusion method.

4. Results and discussion

4.1 The performance of the model

In the context of Android Malware Detection, the performance of various machine learning models was evaluated using a dataset with two classes shown in **Table 1**, **Figure 3**, **Figure 4**, **Figure 5** and **Figure 6**. The classes represent benign and malicious Android applications. This analysis aimed to determine which model could most effectively distinguish between the two. The Logistic Regression model achieved an accuracy of 70% in identifying benign and malicious applications. The confusion matrix shows that it correctly identified 1324 benign apps as benign and 313 malicious apps as malicious. However, it misclassified 626 malicious apps as benign and 91 benign apps as malicious. The relatively high number of false negatives indicates that while logistic regression is somewhat effective, it might not be sufficiently sensitive to all malware patterns without further tuning or feature engineering. The Decision Tree model showed an improvement, achieving an accuracy of 74%. According to its confusion matrix, 1314 benign apps and 439 malicious apps were correctly classified. However, 500 malicious apps were incorrectly labeled as benign, and 101 benign apps were mistakenly classified as malicious. This model offers better sensitivity than logistic regression but still struggles with a substantial number of false negatives. The KNN model demonstrated significant improvement with an accuracy of 84%. It correctly identified 1235 benign apps and 754 malicious apps, with fewer false negatives (185) and false positives (180) compared to the previous models. KNN's higher accuracy suggests it is better at capturing the complexities and nuances in the data that distinguish between the app types.

A fusion approach combining Logistic Regression, Decision Tree, and K-Nearest Neighbors was implemented to leverage the strengths of individual models. This method achieved the highest accuracy of 88%. The confusion matrix for this approach indicates a robust performance with 1304 benign apps correctly identified and 773 malicious apps accurately detected. The reduction in false negatives (166) and false positives (111) demonstrates that integrating multiple models provides a more balanced approach, effectively increasing sensitivity

and specificity. The integration of diverse modeling techniques in the proposed fusion method appears to mitigate individual model weaknesses, leading to more accurate malware detection. This methodology not only highlights the importance of model selection in cybersecurity tasks but also underscores the potential of ensemble techniques in enhancing prediction accuracy. Thus, this approach could be highly beneficial for developing robust Android malware detection systems.

Table 1. The performance of different models in testing dataset.

Model Name	Testing Accuracy
LR	0.70
DT	0.74
KNN	0.84
Proposed model fusion method (LR+DT+KNN)	0.88

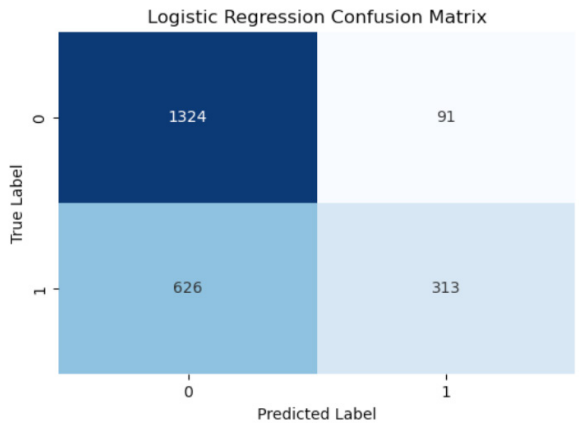

Figure 3. The confusion matrix based on the logistic regression.

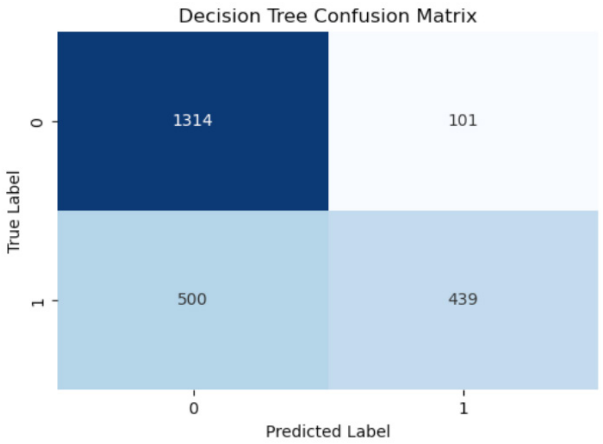

Figure 4. The confusion matrix based on the decision tree.

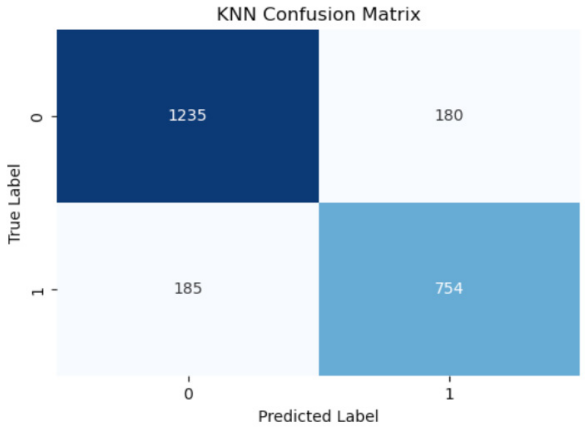

Figure 5. The confusion matrix based on the KNN.

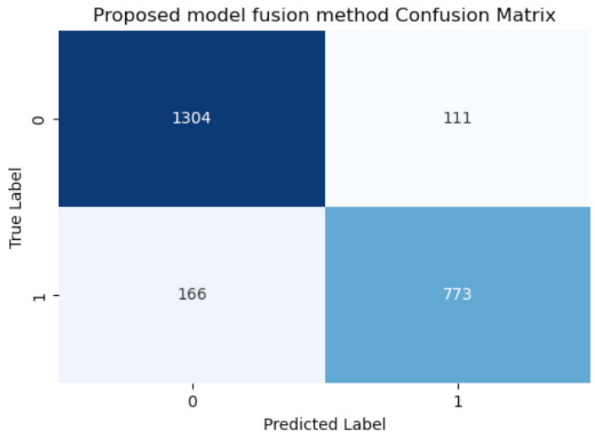

Figure 6. The confusion matrix based on the proposed model fusion method.

Figure 7 illustrates the impact of varying the number of estimators in Random Forest model in the performance of model fusion method for a given dataset. The accuracy increases sharply as the number of estimators rises from 1 to around 5. This suggests that the model benefits significantly from an initial increase in the number of trees, which helps to reduce variance and improve model stability. Between 5 and 15 estimators, accuracy shows some fluctuations, peaking at around 10 and 15 estimators. This variation indicates that certain configurations of the forest are more optimal for this specific dataset, capturing the underlying patterns more effectively than others. After reaching approximately 10 estimators, the accuracy tends to plateau, with slight increases and decreases but generally maintaining a high level of performance. This plateau suggests that adding more estimators beyond this point does not contrib-

ute significantly to improving accuracy, indicating a point of diminishing returns. The highest observed accuracy is just under 0.88, demonstrating that the Random Forest model, when properly tuned with the right number of estimators, can achieve a high degree of predictive accuracy. This analysis is crucial for understanding how the complexity of a Random Forest model affects its performance. By tuning the number of estimators, one can balance the trade-off between training time and model accuracy. The findings suggest that for this particular dataset and fusion method, using around 10 to 15 estimators might offer an optimal balance.

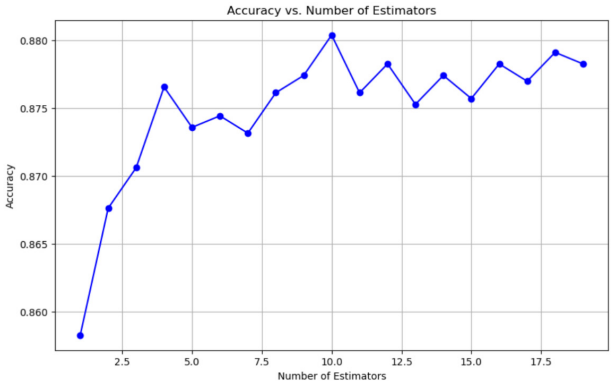

Figure 7. The influence of number of estimators in the performance of model fusion method.

4.2 Discussion

The experimental results provide valuable insights into the efficacy and limitations of the proposed multi-model fusion approach for Android malware detection. While the fusion method demonstrates significant improvements in detection accuracy compared to individual models, several important aspects warrant discussion. Although the fusion approach yields higher accuracy, it is essential to acknowledge that it also introduces additional complexity to the detection system. Integrating multiple machine learning models requires careful consideration of computational resources and model interpretability. The computational overhead associated with training and deploying a fused model may pose challenges, particularly in resource-constrained environments such as mobile devices. Using more advanced hardware may solve this issue [40–43]. Moreover, the interpretability of the fused model may be compromised, making it challenging to understand the underlying decision-making process and identify potential vulnerabilities or biases. Furthermore, while the fusion approach enhances detection accuracy, it may not completely eliminate the risk of false positives and false negatives. Despite achieving a high level of accuracy, the fused model still exhibits misclassifications, as evidenced by the confusion matrix analysis. False positives can lead to unnecessary alerts or actions, while false negatives can result in undetected malware threats, both of which have significant implications for system security and usability. Additionally, the choice of machine learning models included in the fusion approach warrants consideration. While logistic regression, decision tree, and K-nearest neighbors are commonly used for classification tasks, they each have inherent strengths and weaknesses. For example, logistic regression may struggle with capturing complex nonlinear relationships, while decision trees are prone to overfitting. By combining these models, the fusion approach attempts to leverage their complementary strengths, but it also inherits their respective limitations. Therefore, exploring alternative or more sophisticated modeling techniques [44,45] may further enhance the effectiveness of the fusion approach.

5. Conclusions

In conclusion, the study highlights the critical importance of robust Android malware detection in the face of escalating security threats in the digital era. Traditional single-model machine learning approaches have demonstrated limitations in effectively capturing the diverse behaviors of malware. To address this challenge, a novel multi-model fusion approach was developed, leveraging the strengths of logistic regression, decision tree, and K-nearest neighbors models. The fusion method exhibited superior performance compared to individual models, achieving higher accuracy and better malware prediction. This indicates that integrating diverse modeling techniques enhances the resilience and accuracy of detection systems, particularly in identifying new

and sophisticated threats. Moreover, the study underscores the significance of ensemble techniques in cybersecurity tasks, providing a pathway for future research in developing more robust Android malware detection systems. By advancing detection methodologies, we can better safeguard users' privacy, financial assets, and critical infrastructure in the ever-evolving landscape of digital security threats.

Conflict of Interest

The authors declare no conflict of interest.

References

[1] Zhou, L., Luo, Z., Pan, X., 2024. Machine learning-based system reliability analysis with Gaussian Process Regression. *arXiv preprint arXiv:2403.11125*.

[2] Pan, X., Luo, Z., Zhou, L., 2024. Navigating the landscape of distributed file systems: Architectures, implementations, and considerations. *arXiv preprint arXiv:2403.15701*.

[3] Qiu, Y., Wang, J., Jin, Z., Chen, H., Zhang, M., Guo, L., 2022. Pose-guided matching based on deep learning for assessing quality of action on rehabilitation training. Biomedical Signal Processing and Control, 72, 103323.

[4] Chen, F., et al., 2024. Comprehensive Survey of Model Compression and Speed up for Vision Transformers. *arXiv preprint arXiv:2404.10407*.

[5] Zhou, L., Wang, M., Zhou, N., 2024. Distributed Federated learning-based deep learning model for privacy MRI brain tumor detection. *arXiv preprint arXiv:2404.10026*.

[6] Zhou, L., Zhang, H., Zhou, N., 2024. Double-compressed artificial neural network for efficient model storage in customer churn prediction. Artificial Intelligence Advances. 6(1), 1-12.

[7] Liu, Y., et al., 2021. Measuring distance using ultra-wideband radio technology enhanced by extreme gradient boosting decision tree (XGBoost). Automation in Construction, 126, 103678.

[8] Liu, Y., Bao, Y., 2023. Real-time remote measurement of distance using ultra-wideband (UWB) sensors. Automation in Construction, 150, 104849.

[9] Arshad, S., et al., 2016. Android malware detection and protection: a survey. International Journal of Advanced Computer Science and Applications, 7(2), 463–475.

[10] Rashidi, B., Fung, C.J., 2015. A survey of android security threats and defenses. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 6(3), 3-35.

[11] Arshad, S., Shah, M. A., Khan, A., Ahmed, M., 2016. Android malware detection and protection: a survey. International Journal of Advanced Computer Science and Applications, 7(2), 463–475.

[12] Rashidi, B., Fung, C.J., 2015. A survey of Android security threats and defenses. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications. 6(3), 3–35.

[13] Li, L., et al., 2017. Static analysis of Android apps: A systematic literature review. Information and Software Technology. 88, 67–95.

[14] Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., Cavallaro, L., 2017. The evolution of Android malware and Android analysis techniques. Computing Surveys. 49(4), 76:1–76:41.

[15] Faruki, P., et al., 2015. Android security: a survey of issues, malware penetration, and defenses. IEEE Communications Surveys and Tutorials. 17(2), 998–1022.

[16] Alqahtani, E.J., Zagrouba, R., Almuhaideb, A., 2019. A survey on Android malware detection techniques using machine learning algorithms. Proceedings of the 6th International Conference on Software Defined Systems. 110–117.

[17] Souri, A., Hosseini, R., 2018. A state-of-the-art survey of malware detection approaches using data mining techniques. Human-centric Computing and Information Sciences. 8(1), 3.

[18] Faruki, P., et al., 2015. Android security: A survey of issues, malware penetration, and defenses. IEEE Communications Surveys and Tutorials. 17(2), 998–1022.

[19] Felt, A. P., Finifter, M., Chin, E., Hanna, S., Wagner, D., 2011. A survey of mobile malware in the wild. In Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM) (pp. 3–14).

[20] Android Security and Privacy. (2018). 2018 Year In Review. [cited April 30, 2020]. Available from https://source.android.com/security/reports/Google_Android_Security_2018_Report_Final.pdf

[21] Zhou, Y., Jiang, X., 2012. Dissecting Android malware: Characterization and evolution. In Proceedings of the IEEE Symposium on Security and Privacy (pp. 95–109).

[22] Protsenko, M., Muller, T., 2014. Android malware detection based on software complexity metrics. In Proceedings of the International Conference on Trust, Privacy & Security in Digital Business (pp. 24–35).

[23] Yang, C., et al., 2014. DroidMiner: Automated mining and characterization of fine-grained malicious behaviors in Android applications. In Proceedings of the European Symposium on Research in Computer Security (ESORICS) (pp. 163–182).

[24] Xu, K., Li, Y., Deng, R.H., 2016. ICCDetector: ICC-based malware detection on Android. IEEE Transactions on Information Forensics and Security. 11(6), 1252–1264.

[25] Yang, M., Wen, Q., 2017. Detecting Android malware by applying classification techniques on images patterns. In Proceedings of the IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA) (pp. 344–347).

[26] Martín, A., Menéndez, H. D., Camacho, D., 2017. MOCDroid: Multiobjective evolutionary classifier for Android malware detection. Soft Computing. 21(24), 7405–7415.

[27] Qiu, Y., et al., 2024. A novel image expression-driven modeling strategy for coke quality prediction in the smart cokemaking process. Energy. 294, 130866.

[28] Zhao, F., et al., 2023. A new method using LLMs for keypoints generation in qualitative data analysis. In 2023 IEEE Conference on Artificial Intelligence (CAI). IEEE.

[29] Liu, Y., Yang, H., Wu, C., 2023. Unveiling patterns: a study on semi-supervised classification of strip surface defects. IEEE Access. 11, 119933-119946.

[30] Li, S., et al., 2024. Application of semi-supervised learning in image classification: research on fusion of labeled and unlabeled data. IEEE Access.

[31] Luo, Z., Xu, H., Chen, F., 2019. Audio Sentiment Analysis by Heterogeneous Signal Features Learned from Utterance-Based Parallel Neural Network. AffCon@ AAAI.

[32] Chen, F., Luo, Z., Xu, Y., et al., 2019. Complementary fusion of multi-features and multi-modalities in sentiment analysis. *arXiv preprint arXiv:1904.08138*.

[33] Luo, Z., Zeng, X, Bao, Z., et al., 2019. Deep learning-based strategy for macromolecules classification with imbalanced data from cellular electron cryotomography. 2019 International Joint Conference on Neural Networks (IJCNN). IEEE.

[34] Luo, Z., 2023. Knowledge-guided aspect-based summarization. 2023 International Conference on Communications, Computing and Artificial Intelligence (CCCAI). IEEE.

[35] Shen, Y., Gu, H.M., Qin, S., Zhang, D.W., 2022. Surf4, cargo trafficking, lipid metabolism, and therapeutic implications. Journal of Molecular Cell Biology. 14(9), mjac063.

[36] Qiu, Y., Chen, H., Dong, X., et al., 2024. IFViT: Interpretable Fixed-Length Representation for Fingerprint Matching via Vision Transformer. *arXiv preprint arXiv:2404.08237*.

[37] Shen, F., Vecchio, J. D., Mohaisen, A., Ko, S. Y., Ziarek, L., 2019. Android malware detection

using complex-flows. IEEE Transactions on Mobile Computing. 18(6), 1231–1245.

[38] Gorla, A., Tavecchia, Ilaria, Gross, Florian, et al., 2014. Checking app behavior against app descriptions. In Proceedings of the 36th International Conference on Software Engineering (ICSE) (pp. 1025–1035).

[39] Li, Y., Y. Ma, M. Chen, et al., 2017. A detecting method for malicious mobile application based on incremental SVM. In Proceedings of the 3rd IEEE International Conference on Computer Communication (ICCC) (pp. 1246–1250).

[40] Deng, X., Oda, S., Kawano, Y., 2023. Graphene-based midinfrared photodetector with bull's eye plasmonic antenna. Optical Engineering. 62(9), 097102-097102.

[41] Sugaya, T., Deng, X., 2019. Resonant frequency tuning of terahertz plasmonic structures based on solid immersion method. 2019 44th International Conference on Infrared, Millimeter, and Terahertz Waves (IRMMW-THz). IEEE.

[42] Deng, X., Li, L., Enomoto, Mitsuhiro, et al., 2019. Continuously frequency-tuneable plasmonic structures for terahertz bio-sensing and spectroscopy. Scientific reports. 9(1), 3498.

[43] Deng, X., Simanullang, M., Kawano, Y., 2018. Ge-core/a-si-shell nanowire-based field-effect transistor for sensitive terahertz detection. Photonics, 5(2).

[44] Li, S., Singh, Kanupriya, Riedelet, Nathan, et al., 2022. Digital learning experience design and research of a self-paced online course for risk-based inspection of food imports. Food Control. 135, 108698.

[45] Yu, F., Milord, J., Orton, Sarah, et al., 2021. Students' evaluation toward online teaching strategies for engineering courses during COVID. 2021 ASEE Midwest Section Conference.