Journal of
Computer & Electrical and Electronics Engineering Sciences
Original Article
DOI: 10.51271/JCEEES-0022

Android malware detection with image-based features using transfer learning methods

Oğuzhan Sezer, İbrahim Alper Doğru, Kazım Kılıç, İsmail Atacak
Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara, Turkiye

Cite this article as: Sezer, O., Doğru, İ.A., Kılıç, K., & Atacak, İ. (2025). Android malware detection with image-based features using transfer learning methods. *J Comp Electr Electron Eng Sci,* 3(1),8-13.

Received: 04.11.2024 ◆ **Accepted**: 02.01.2025 ◆ **Published**: 17.04.2025

ABSTRACT

The increase in mobile device usage today has also brought security threats to Android devices. Ensuring the security of Android devices is crucial for protecting user data. Numerous studies have been conducted that encompass both traditional and AI-based methods to secure Android devices. This study aims to enrich the literature by evaluating and comparing the performance of three transfer learning models (ResNet50, DenseNet201, and VGG16) on a hybrid dataset adapted with specific parameters. The dataset obtained from AndroZoo and Drebin classifies the data as benign and malicious for effective malware detection. This research used grayscale images from the dataset to train the aforementioned transfer learning models. The results show that transfer learning models can be successfully used in malware detection. It has been observed that the VGG16 model achieved the best performance in malware detection with an accuracy of 97.24%, a precision of 97.26%, a recall of 97.24%, and an Auc value of 99.33%. These findings may also provide valuable contributions to developing mobile security applications.
Keywords: Mobile Security, Android Malware Detection, Transfer Learning, Mobile App Security, Deep Learning

INTRODUCTION

The rapid increase in internet-connected devices has made cybersecurity increasingly crucial for protecting digital assets from cyberattacks. Over time, the number of devices connected to the Internet has increased significantly and has affected various industries. Organizations and critical sectors equipped with technologies such as the Internet of Things (IoT), healthcare and medical devices, and Internet banking have found broader access opportunities through the Internet. In addition, a wide range of users, such as educational institutions, businesses, and banking, benefit from the services offered over the Internet (Kumar and Panda, 2023; Almomani et al., 2022).

Due to its openness and popularity, the Android operating system is the primary target for attacks on smartphones, accounting for approximately 98.5% of such attacks, with malware being the primary vector. The most worrying thing is that users access Internet services without understanding the impact of cyber threats on the Internet (Kumar and Panda, 2023; Almomani et al., 2022). The services provided by the Internet affect our daily lives. As cyber threats evolve, the situation may become dangerous or even life-threatening (Kumar and Panda, 2023). By 2023, about 5.44 billion users have had access to the Internet. Statistics show that the Internet will continue to transform people's daily lives by providing services that are easily accessible. In 2021, more than half of the people in the world, about 5 billion people, had access to the Internet (Kumar and Panda, 2023).

Static and dynamic analysis methods are generally used to detect Android malware. Static analysis examines the source code of malware without running it (Almomani et al., 2022), and while cost-effective, it can be insufficient to identify zero-day attacks and obfuscated malware (Almomani et al., 2022; Ban et al., 2022). Dynamic analysis observes the real-time malware behaviors in isolated virtual machines; however, this method may have limitations as the malware can alter its behavior in response (Almomani et al., 2022).

Using traditional machine learning methods, the researchers combined static and dynamic analysis features used in malware detection to aim for higher accuracy values. However, these methods are time-consuming and complex, often requiring manual feature selection (Ban et al., 2022). Deep learning methods have demonstrated superior accuracy in malware detection by eliminating the need for manual feature selection. These methods leverage various features, including API calls and permissions, to effectively identify malicious software (Ban et al., 2022).

In recent years, deep learning methods have increasingly been used in malware detection. Image-based models have often been preferred because they achieve high accuracy in malware detection. These models allow us to study the behavior of malware in the file structure (Kumar and Panda, 2023; Almomani et al., 2022; Ban et al., 2022; Zou et al., 2022). The primary problems addressed in this study are the increasing threat level of Android malware, the inadequacies

Corresponding Author: Oğuzhan Sezer, oguzhansezer@gazi.edu.tr

of existing detection methods, and the lack of frequent joint usage of datasets in the literature. This study aims to apply three different transfer learning methods to a hybrid dataset constructed with specific parameters. The rationale behind employing a hybrid dataset is to create a unique and distinct dataset. The limited use of hybrid datasets is highlighted as an advantage in the literature, as it allows for creating a unique dataset. Additionally, merging multiple datasets is considered a factor that reduces overfitting. Another significant advantage is that, even if one dataset contains corrupted or missing data, the hybrid dataset mitigates such issues, ensuring data integrity.

Six thousand data were used from the Androzoo dataset. In addition, 5100 data were planned to be used from the Drebin dataset. However, 42 data were lost due to errors in the creation of the Drebin dataset. A total of 11058 data were obtained. Malware detection was performed on three different transfer learning models using grayscale images from the hybrid dataset. The results of these models were compared with similar studies in the literature, and it was determined which model worked best on this new dataset.

The main contributions of this study are as follows:

- **Use of a hybrid dataset:** By leveraging the advantages of hybrid datasets rarely utilized in the literature, a balanced and unique dataset was created by combining the Androzoo and Drebin datasets. This approach offers benefits such as preventing data loss and reducing the risk of overfitting.
- **Utilization of grayscale images:** Grayscale images were used for Android malware detection, and this method enabled the evaluation of the performance of different transfer learning models. By converting the images into pixel matrices, the behavioral patterns of malware were effectively analyzed.
- **Comparison of three transfer learning models:** Widely used transfer learning models, including ResNet50, DenseNet201, and VGG16, were compared on the same dataset, and their impact on Android malware detection was investigated. This comparison revealed which model achieved higher accuracy.
- **Malware detection using static analysis methods:** The study focused on malware detection through static analysis methods and demonstrated that these methods achieve high accuracy rates.

METHODS

Androzoo is a dataset that works on mobile devices with the Android operating system. This dataset was created so that applications can be analyzed. It is also frequently used in mobile security research. Androzoo contains a huge number of Android app examples. In this way, it is beneficial to researchers. Samples on Androzoo have been obtained in various ways from the Google Play Store and other sources. The dataset contains many unique features. In this way, they have made great contributions to mobile security analysis (Allix et al., 2016). In the AndroZoo dataset, filtering was applied to select APK files that meet specific criteria for analysis. APK files with a VirusTotal detection rate of 10 or more were considered for the malicious dataset. Additionally, the analysis focused exclusively on APKs smaller than 2 MB, targeting less complex and smaller-sized files. Lastly, APKs sourced from the Google Play Store were selected, limiting the scope to applications from this specific marketplace. For the benign dataset, APKs with a VirusTotal detection rate of zero were utilized. Similarly, APKs smaller than 2 MB and obtained from the Google Play Store were chosen to ensure security and consistency.

Drebin was developed to detect Android malware. Performs analysis using static analysis methods. It is not necessary to run Android applications while using Drebin. Drebin makes classifications based on application characteristics. In this way, threats are detected, and reports are generated. Drebin is used in areas such as malware detection and security analysis (Arp et al., 2014). The APK files included in the dataset predominantly represent malicious software and are categorized based on specific behavioral traits. The APKs were collected from both official sources and third-party application stores. The files in the dataset are generally smaller than 10 MB, enabling fast and efficient analysis. The APKs were labeled as either malicious or benign using tools like VirusTotal and analyzed based on their behaviors.

A total of 11058 data obtained from Drebin and Androzoo datasets were used in this study. This dataset is divided into 5529 benign APKs and approximately 5529 malignant APKs. APKs created for benign and malignant were obtained from both Drebin and Androzoo datasets. The large number of samples in the data set made the study more efficient. Two different classes were used in the data set: benign and malignant. 70% of the dataset was used for training, 20% for testing, and 10% for validation.

The datasets used in this study, Androzoo, and Drebin, comprise approximately 5,529 benign and 5,529 malware samples, making it a balanced dataset. The 11,058 APK files that were obtained were converted into Classes.dex files. From the converted files, the classes.dex file was further transformed to obtain byte files. These byte files were converted into pixel matrices ranging between 0-255. This process was applied to transform the image data into pixels. Finally, these pixel matrices were used to obtain grayscale images. Thus, the data obtained from the APK files were converted into grayscale data to be used for Android malware detection. The dimensions of the grayscale images are set to 255*255.

In this study, the TensorFlow library, commonly used in deep learning, was utilized. Data augmentation was performed using the ImageGenerator function in TensorFlow. ImageGenerator does not physically augment the dataset. Data augmentation was applied to the training dataset, which was split by 70%. Diversifying image data helped the model learn more generalized features. Transformations such as flipping, rotating, and zooming images allowed the model to learn from different perspectives. Additionally, data augmentation increased the diversity of the training dataset, reducing the risk of overfitting. Finally, results were obtained by training with three different transfer learning models. These trained models were compared.

Figure 1 shows the overall architecture of the model used in this study.

Transfer learning is the process of applying the experiences obtained by a trained model to a new problem. Here, three different transfer learning methods are examined.

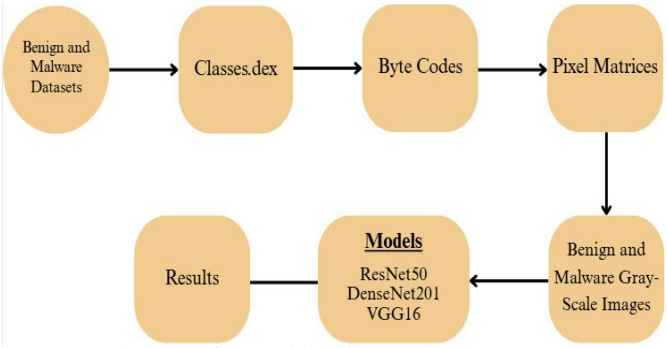

Figure 1. Architecture of the model used

ResNet50 is a convolutional neural network (CNN) model developed by Microsoft Research. This transfer learning model uses an approach known as "skip connection" (Kumar and Panda, 2023). This model consists of 50 layers. These layer blocks include convolutional layers and ReLU activation functions (Mukti and Biswas, 2019). The "skip connection" structure allows the flow of information to switch between layers. Resnet50 is frequently used in tasks such as object recognition and classification. (Mukti and Biswas, 2019; Abdulazeez et al., 2024). ResNet50 is a pre-trained CNN model that incorporates a 50-layer deep architecture with residual connections. The final layers of the model were adapted to address the classification problem in this study, generating outputs for two classes: benign and malware. The output layer consists of two fully connected layers followed by a softmax activation function. To mitigate overfitting, dropout rates of 0.25 and 0.5 were applied, respectively.

The Adam optimization algorithm was utilized during training. The Adam optimizer adjusts momentum and adaptive learning rates with beta values set to 0.9 and 0.99, respectively. The weight decay value was set to 0.01 to regularize the model's parameters. The model loss was calculated using the cross entropy loss function.

The batch size during training was set to 64. The learning rate was defined as 0.001, as recommended by FastAI, and a one-cycle learning rate scheduling strategy was adopted. This approach allows for an efficient and dynamic adjustment of the learning rate throughout the training process, improving convergence and generalization. The model was trained for 50 epochs.

DenseNet201 is a convolutional neural network model. This model contains 201 layers. It uses the output of each layer as an input to all the layers after it. In this way, it ensures efficiency (Pramesti et al., 2023). This structure facilitates deep learning by improving the flow of information. It also allows the model to represent the feature map with a small number of parameters. These features enable DenseNet201 to achieve high accuracy rates in training deep networks (Altaf et al., 2023). The final layers of the DenseNet201 model were restructured to suit the classification problem addressed in this study. The model was tailored for two classes (benign and malware), with the output layer comprising two fully connected layers followed by a softmax activation function. To prevent overfitting, dropout rates of 0.25 and 0.5 were applied sequentially.

The Adam optimization algorithm was employed during training, which adjusts momentum and adaptive learning rates with beta values set to 0.9 and 0.99. The model loss was calculated using the Cross-Entropy Loss function to minimize the discrepancy between the classes. The batch size used during the training process was set to 64.

The initial learning rate was defined as 0.001. The ReduceLROnPlateau method was utilized to adjust the learning rate dynamically. This method reduces the learning rate by a factor of 0.2 if the accuracy metric does not improve for five consecutive epochs. The model was trained for a total of 50 epochs.

The VGG16 model is a 16-layer convolutional neural network. These layers consist of 13 convolutional and three fully connected layers. Convolutional layers learn features from images using 3x3 filters. Max pooling layers, on the other hand, reduce the size of feature maps, reducing computational cost. The model classifies through fully connected layers using flattened feature maps. In addition, thanks to the ReLU activation functions, learning efficiency increases (Kumar and Panda, 2023). VGG16 with batch normalization is a pre-trained CNN model that enhances stability during training. The final layers of the model were restructured to address the binary classification problem between benign and malware classes. The output layer consists of two fully connected layers and a softmax activation function, which generates probabilities for the two target classes.

The Adam optimization algorithm was utilized for model optimization, effectively handling momentum and adaptive learning rate adjustments. The beta values for optimization were set to 0.9 and 0.99. To regularize the model, the weight decay parameter was set to 1e-3 (0.001). The model loss was computed using the cross-entropy loss function, which is suitable for minimizing the discrepancy between the predicted and actual class labels in classification tasks.

During training, a batch size of 64 was chosen to balance computational efficiency and convergence performance. The learning rate was initialized with the min_grad_lr parameter, which was calculated during the learning rate finder step. This method ensures an optimal starting learning rate for training. Dynamic learning rate adjustments were implemented using the ReduceLROnPlateau method, which reduces the learning rate by a factor of 0.2 when the accuracy metric fails to improve.

The model was trained for a total of 50 epochs using a one-cycle learning policy, facilitating efficient training and convergence.

RESULTS

The experiment was conducted on a server with an Intel® 12700K CPU running at 2.40 GHz, an NVIDIA GeForce RTX 3070 Ti GPU, and 64 GB of RAM. Jupyter Notebook was used as part of the Python code editor for development. Several commonly used Python packages, such as vision and callbacks, were used to process large data volumes. Fastai libraries were employed for deep-learning image processing. The graphical representation of performance metrics was done using visualization tools like Matplotlib and Seaborn.

Accuracy is the ratio of the samples that the model correctly predicted to the total number of predicted samples. Recall is the ratio of the number of samples that the model correctly identifies as positive to the total number of true positive samples. Precision is the ratio of the model's true positive

predictions to its total positive predictions (Kumar and Panda, 2023).

- **TP (true positive):** True positives are examples that are correctly identified as malware (Kumar and Panda, 2023).
- **TN (true negative):** True negatives are examples that are correctly identified as harmless (Kumar and Panda, 2023).
- **FP (false positive):** False positives are examples defined as malware even though they are harmless (Kumar and Panda, 2023).
- **FN (false negative):** False negatives are examples defined as harmless even though they are malware (Kumar and Panda, 2023).

$$\text{Accuracy}=\frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$\text{Recall}=\frac{TP}{TP+FN} \quad (2)$$

$$\text{Precision}=\frac{TP}{TP+FP} \quad (3)$$

Image-based detection is a method in computer security that analyzes visual content to detect malware or security threats. This approach focuses on application icons, interface elements, screenshots, and other visual components to identify potential risks (Zou et al., 2022).

The accuracy graph of the ResNet50 model is shown in **Figure 2**.

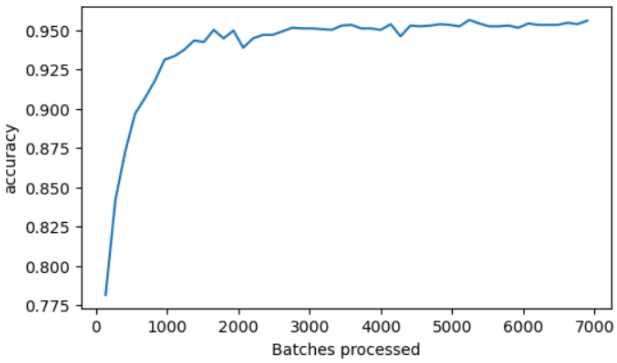

Figure 2. Accuracy graph of the ResNet50 model

Figure 2 illustrates that as iterations on the training set progress, there is an initial sharp increase, indicating that the model enters a rapid learning phase. After approximately 3000 iterations, the accuracy rate stabilizes at a high level. This demonstrates the effectiveness of the model's overall learning capacity and the success of the training process.

The confusion matrix values of the ResNet50 model are shown in **Figure 3**.

Figure 3 shows that the model detects malicious and benign software with generally high accuracy; it makes 1090 true positive and 1025 true negative predictions, with only a small number of false positives (66) and false negatives (30).

The accuracy graph of the DenseNet201 model is shown in **Figure 4**.

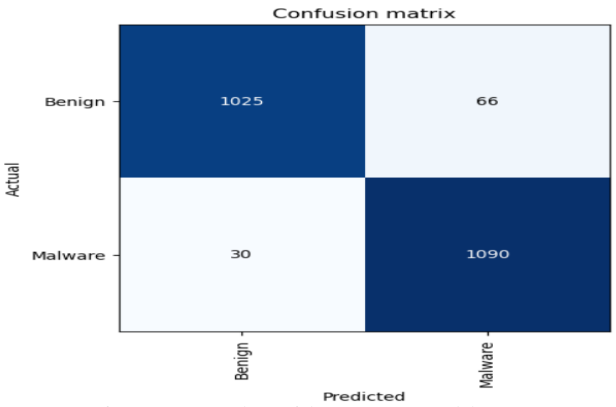

Figure 3. Confusion matrix values of the ResNet50 model

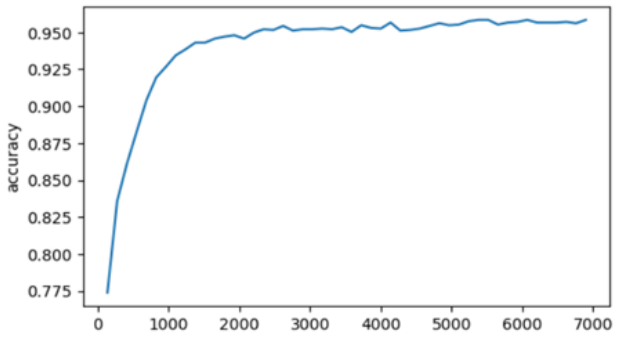

Figure 4. Accuracy graph of the DenseNet201 model

Figure 4 reflects an effective learning process with a rapidly increasing accuracy rate at the beginning of the training, demonstrating stable performance with approximately 95% accuracy after 3000 iterations.

The confusion matrix values of the DenseNet201 model are shown in **Figure 5**.

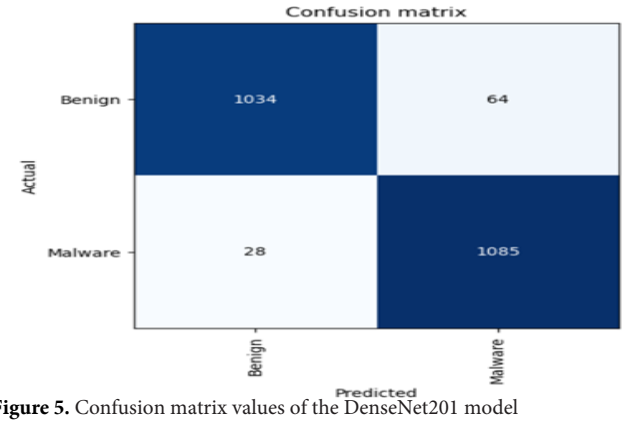

Figure 5. Confusion matrix values of the DenseNet201 model

Figure 5 shows that the model distinguishes between malicious and benign software with generally high accuracy. The model made 1085 true positive and 1034 true negative predictions, with only 64 false positive and 28 false negative errors. These results demonstrate that the model can effectively recognize malicious and benign software.

The accuracy graph of the VGG16 model is shown in **Figure 6**.

Figure 6 shows that during the training process, the model initially exhibits rapid improvement, surpassing 90% accuracy, and achieves values between 96% and 98% after approximately 4000 iterations. This indicates that the model reaches high performance relatively early and maintains this performance throughout the training, highlighting its stable and reliable learning dynamics.

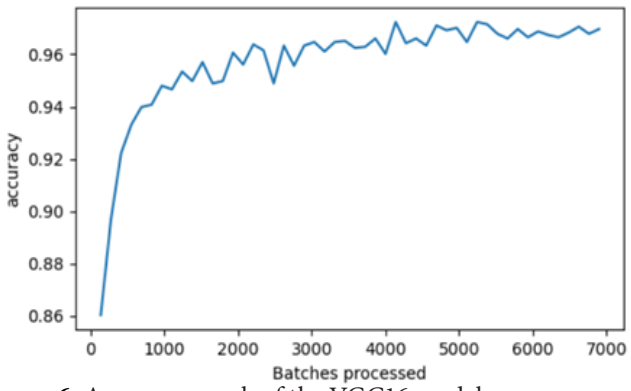

Figure 6. Accuracy graph of the VGG16 model

The confusion matrix values of the VGG16 model are shown in Figure 7.

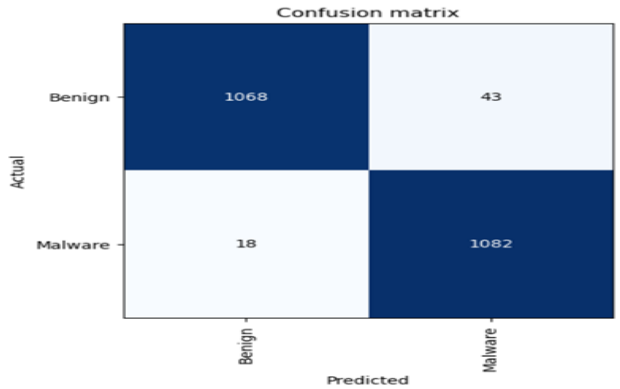

Figure 7. Confusion matrix values of the VGG16 model

Figure 7 demonstrates that the model is highly effective in distinguishing between malicious and benign software. The model successfully classified malicious software with 1082 true positive predictions and benign software with 1068 true negative predictions, making only 43 false positive and 18 false negative errors. These results indicate that the model can detect malicious and benign software with high accuracy and perform reliably in these tasks.

Table 1 presents the performance results of three deep-learning models for Android malware detection. The ResNet50 model shows balanced performance with 95.65% accuracy, 95.72% precision, and 95.63% recall, while its Auroc value of 98.69% indicates high success. DenseNet201 delivers similar results to ResNet50 with 95.83% accuracy and 95.89% precision, demonstrating high sensitivity in detecting malware with 95.82% recall and a remarkable ability to handle misleading examples with a 98.98% Auroc value. The VGG16 model offers more precise results than the other two models, with 97.24% accuracy and 97.26% precision, effectively detecting malware with a 97.24% recall and exhibiting exceptionally high classification performance with a 99.33% Auroc value. Each model possesses strong predictive capabilities despite deceptive or challenging examples, demonstrating their proficiency in Android malware detection.

Table 1. Overall results of the trained models

	Evaluation metrics			
Trained models	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**AUC (%)**
Resnet50	95.65	95.72	95.63	98.69
DenseNet201	95.83	95.89	95.82	98.98
VGG16	97.24	97.26	97.24	99.33

AUC: Area under curve, ResNet50, DenseNet201, and VGG16: Three transfer learning models

Table 2 presents various studies in Android malware detection, comparing the models used, datasets, analysis methods, and the accuracy rates achieved. The MSAE and SHLMD models used static analysis with the MUDFLOW and VirusShare datasets to achieve 94.46% accuracy (Zhu et al., 2022). In comparison, the MADRF-CNN model achieved 96.9% accuracy using the Google Play Store and VirusShare datasets (Zhu et al., 2023). A group of transfer learning models reached an accuracy rate of 95.83% using static analysis on the CICInvesAndMal20 dataset (Ksibi et al., 2024). The Bi-LSTM model achieved an accuracy of 94.12% using dynamic analysis on the KISA-data challenge 2019-Malware dataset (Jeon et al., 2022). The GuardDroid model achieved an accuracy of 91.74% using both dynamic and static analysis on the MUDFLOW dataset (Vasu et al., 2023).

Limitations

In this study, the VGG16 model achieved 97.24% accuracy with static analysis on the Androzoo and Drebin datasets, while the DenseNet201 and ResNet50 models reached 95.83% and 95.65%, respectively. Each of these studies highlights the interaction of different datasets and analysis methods for malware detection, emphasizing the diversity of the field. This comparative table is a valuable resource for evaluating the advantages and limitations of different approaches.

CONCLUSION

This research examines deep learning-based models as effective tools for combating malware threats on the Android platform. With the increasing accessibility of the Internet and

Table 2. Comparison of related works

		Compared features		
Studies	**Models**	**Datasets**	**Method**	**Accuracy (%)**
Zhu et al., 2022.	MSAE, SHLMD	MUDFLOW, VirusShare	Static	94.46
Zhu et al., 2023.	MADRF-CNN	Google Play Store and VirusShare	Static	96.9
Ksibi et al., 2024.	DenseNet169, Xception, InceptionV3, ResNet50 and VGG16	CICInvesandMal2019	Static	95.83
Jeon et al., 2022.	Bi-LSTM	KISA-data challenge 2019-Malware	Static dynamic	94.12
Vasu et al., 2023.	GuardDroid	MUDFLOW	Dynamic static	91.74
This study	VGG16	Androzoo and Drebin	Static	97.24
	DenseNet201	Androzoo and Drebin	Static	95.83
	Resnet50	Androzoo and Drebin	Static	95.65

MSAE: Mean squared approximation error, ResNet50, DenseNet201, and VGG16: Three transfer learning models, Bi-LSTM: Bidirectional long short-term memory

the integration of various functionalities, mobile devices have become attractive targets for cybercriminals. In this context, deep learning models such as ResNet50, DenseNet201, and VGG16 were evaluated through static analysis using the AndroZoo and Drebin datasets. Transfer learning models demonstrated high accuracy, precision, and recall rates, proving their effectiveness in identifying such threats. Notably, the VGG16 model outperformed others with an accuracy of 97.24% and an AUC score of 99.33%, highlighting its superior performance in classifying malware and distinguishing deceptive samples. This study achieved better results than previous works and partially addressed the challenge of hybrid datasets, which have been underrepresented in the literature. In conclusion, this study underscores the potential of deep learning models for Android malware detection and reveals how different datasets can lead to varying outcomes. Additionally, the strengths and limitations of existing methodologies were highlighted by evaluating the proposed models within a comparative framework against other studies in the literature. Future research should focus on enhancing these models' performance by incorporating more datasets and advanced analytical methods. Hybrid models should also be utilized alongside hybrid datasets. Specifically, a combination of transfer learning methods, machine learning algorithms, and attention mechanisms can facilitate the development of diverse hybrid models. Furthermore, in addition to image processing, audio-based analyses should also be explored. Such advancements will strengthen Android security solutions, ultimately aiding end-users and organizations in mitigating malware threats.

ETHICAL DECLARATIONS

Referee Evaluation Process

Externally peer-reviewed.

Conflict of Interest Statement

The authors have no conflicts of interest to declare.

Financial Disclosure

The authors declared that this study has received no financial support.

Author Contributions

All of the authors declare that they have all participated in the design, execution, and analysis of the paper, and that they have approved the final version.

REFERENCES

Abdulazeez, F.A., Ahmed, I.T., & Hammad, B.T. (2024). Examining the performance of various pretrained convolutional neural network models in malware detection. *Appl Sci*, 14(6),2614. doi:10.3390/app14062614

Allix, K., Bissyandé, T.F., Klein, J., & Le Traon, Y. (2016, May). Androzoo: collecting millions of android apps for the research community. In Proceedings of the 13th international conference on mining software repositories (pp. 468-471).

Almomani, I., Alkhayer, A., & El-Shafai, W. (2022). An automated vision-based deep learning model for efficient detection of android malware attacks. *IEEE Access*, 10,2700-2720. doi:10.1109/ACCESS.2022.3140341

Altaf, Y., Wahid, A., & Kirmani, M.M. (2023, February). Deep learning approach for sign language recognition using densenet201 with transfer learning. In 2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS) (pp. 1-6). IEEE.

Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C.E.R.T. (2014, February). Drebin: effective and explainable detection of android malware in your pocket. In Ndss (Vol. 14, No. 1, pp. 23-26).

Ban, Y., Lee, S., Song, D., Cho, H., & Yi, J.H. (2022). Fam: featuring android malware for deep learning-based familial analysis. *IEEE Access*, 10, 20008-20018. doi:10.1109/ACCESS.2022.3151357

Chaganti, R., Ravi, V., & Pham, T.D. (2022). Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification. *J Informat Security Appl*, 69,103306. doi:10.1016/j.jisa.2022.103306

Chen, S., Lang, B., Liu, H., Chen, Y., & Song, Y. (2024). Android malware detection method based on graph attention networks and deep fusion of multimodal features. *Expert Systems Appl*, 237,121617. doi:10.1016/j.eswa.2023.121617

Jeon, J., Jeong, B., Baek, S., & Jeong, Y.S. (2021). Hybrid malware detection based on Bi-LSTM and SPP-Net for smart IoT. *IEEE Transact Industr Informat,* 18(7),4830-4837. doi:10.1109/TII.2021.3119778

Ksibi, A., Zakariah, M., Almuqren, L., & Alluhaidan, A.S. (2024). Efficient android malware identification with limited training data utilizing multiple convolution neural network techniques. *Eng Appl Artific Intell*, 127,107390. doi:10.1016/j.engappai.2023.107390

Kumar, S., & Panda, K. (2023). SDIF-CNN: stacking deep image features using fine-tuned convolution neural network models for real-world malware detection and classification. *Appl Soft Comput*, 146,110676. doi:10.1016/j.asoc.2023.110676

Mukti, I.Z., & Biswas, D. (2019, December). Transfer learning based plant diseases detection using ResNet50. In 2019 4th International conference on electrical information and communication technology (EICT) (pp. 1-6). IEEE.

Pramesti, E., Mutiara, A.B., & Refianti, R. (2023, December). Implementation of Deep Learning using convolutional neural network for skin disease classification with DenseNet-201 architecture. In 2023 Eighth International Conference on Informatics and Computing (ICIC) (pp. 1-6). IEEE.

Vasu, T., Fiza, S., Kumar, A.K., Devi, V.S., Kumar, C.N., & Kubra, A. (2023). Improved chimp optimization algorithm (ICOA) feature selection and deep neural network framework for internet of things (IoT) based android malware detection. *Measur Sensors*, 28,100785. doi:10.1016/j.measen.2023.100785

Yang, S., Wang, Y., Xu, H., Xu, F., & Chen, M. (2022). An android malware detection and classification approach based on contrastive lerning. *Comput Securit*, 123,102915. doi:10.1016/j.cose.2022.102915

Zhu, H., Wei, H., Wang, L., Xu, Z., & Sheng, V.S. (2023). An effective end-to-end android malware detection method. *Expert Syst Appl*, 218,119593. doi:10.1016/j.eswa.2023.119593

Zhu, H.J., Wang, L.M., Zhong, S., Li, Y., & Sheng, V.S. (2021). A hybrid deep network framework for android malware detection. *IEEE Transact Knowl Data Eng*, 34(12),5558-5570. doi:10.1109/TKDE.2021.3067658

Zou, B., Cao, C., Tao, F., & Wang, L. (2022). IMCLNet: a lightweight deep neural network for ımage-based malware classification. *J Informat Security Appl,* 70,103313. doi:10.1016/j.jisa.2022.103313