

Original Article

Android Malware Detection using Deep Learning based Feature Fusion and Ensemble Learning Models through Image Representations

G.Srinivas¹, M.V. Rajesh²

¹Department of Computer Science and Engineering, Aditya University, Surampalem, Andhra Pradesh, India.

²Department of IT, Aditya University, Surampalem, Andhra Pradesh, India.

¹Corresponding Author : srinivascse531@gmail.com

Received: 08 June 2026

Revised: 19 August 2026

Accepted: 26 August 2026

Published: 30 September 2026

Abstract - Malware detection has become a serious task because of rapid development of malicious software and its evolving patterns. This paper presents a comprehensive framework for malware classification by integrating deep learning, handcrafted feature extraction and machine learning techniques. Initially, a CNN is employed as a baseline model, followed by transfer learning models, including DenseNet and MobileNet, to enhance feature representation. To further boost performance, several handcrafted and deep features, including color histogram, Gabor, GLCM, HOG, LBP, ORB, SIFT, wavelet transform features and pre-trained deep features are extracted and combined to create a hybrid feature set. These features are then used to train a number of ML classifiers such as Random Forest, SVM, Logistic Regression, XGBoost, AdaBoost, Gradient Boosting, LightGBM and CatBoost. Among these, XGBoost achieved superior performance among individual models. Moreover, ensemble methods like voting and stacking classifiers are used to boost robustness and accuracy. The stacking classifier utilizing the metaclassifier Random Forest gave the best results of 91.6% accuracy. The results highlight the effectiveness and scalability of the proposed hybrid feature-based ensemble approach, which improves the performance of standalone deep learning models in the field of malware detection.

Keywords - Android malware detection, Deep learning, Ensemble learning, Feature fusion, Image-based analysis.

1. Introduction

With the growth of Android applications, there has been a continuous rise of malicious applications trying to compromise the privacy, security and integrity of user data and devices. Android malware can use various methods to hide its malicious activity, and can change over time, making it difficult to detect through signature-based detection against malware that has not been previously encountered or modified. These challenges have increased the need for automated detection techniques capable of identifying discriminative patterns from Android application data and classifying apps as benign or malicious.

ML & DL techniques have therefore been increasingly investigated for malware detection. Conventional machine learning approaches generally depend on manually selected features such as permissions, API calls and other static application features. Deep learning approaches, in contrast, can learn feature representations directly from the input data. In particular, image-based malware analysis has received attention because binary or DEX data can be transformed into grayscale images, allowing image-processing and DL

techniques to identify structural and spatial patterns. CNNs can learn such spatial patterns directly from malware images, while transfer learning models can provide richer feature representations by utilizing pretrained architectures. However, a single deep representation may not capture all the complementary texture, structural, statistical and local patterns present in DEX image representations.

Handcrafted image descriptors provide an alternative source of complementary information. Descriptors such as HOG, LBP, GLCM, Gabor, SIFT, and ORB can represent different characteristics of an image, including edge distributions, local texture, spatial relationships, frequency information, and distinctive key points.

Combining such descriptors with deep features can provide a more diverse representation than using either feature category independently. In addition, ML classifiers can provide different decision boundaries for the resulting high-dimensional feature representation, while ensemble methods can combine the predictions of complementary classifiers to improve classification stability.



Although previous studies have explored deep learning, handcrafted feature extraction, feature selection, feature fusion and ensemble learning, these approaches have generally focused on specific representations or combinations of techniques. The reviewed literature also indicates that image-based malware detection has mainly emphasized deep learning representations, while feature-based approaches have investigated selected static or handcrafted characteristics. Therefore, a research gap remains in developing and evaluating a unified framework that combines DEX image representations, diverse handcrafted image descriptors, pretrained deep feature representations, dimensionality reduction, multiple ML classifiers and ensemble learning for Android malware classification. In particular, the effectiveness of combining complementary handcrafted and deep features and subsequently exploiting the fused representation through voting and stacking classifiers requires further investigation.

In order to fill this, an image-based Android malware classification framework is developed by obtaining classes. dex representation from Android APK files. The extracted DEX data are converted to grayscale images and then preprocess before it is classified. The first CNN baseline model is tested, and then the transfer learning models are applied to study the deep representation efficacy. After that, image descriptors are created manually, and deep features are extracted from pre-trained models using feature-level fusion. PCA is used to standardize and reduce the feature representation to a high dimension, keeping 95% of the variation. Subsequent, various ML classifiers are tested with the reduced feature representation. Lastly, voting and stacking ensemble methods are implemented with base classifiers of Support Vector Machine (SVM), XGBoost and LightGBM; the meta-learner of the stacking model is Random Forest.

The major contributions of this work are as follows:

- Development of a baseline CNN and transfer learning models for malware classification.
- Extraction and integration of diverse handcrafted and deep features to form a hybrid feature representation.
- Comprehensive evaluation of multiple ML classifiers on the fused feature set.
- Implementation of ensemble learning models to increase classification accuracy.

Results of experiments show that the proposed hybrid feature-based ensemble approach improves the malware detection performance as compared to individual models.

2. Literature Survey

In order to find the effectiveness in the detection of Android malware, the work done by Anuradha et al. [1] improved the detection using a hybrid DL model. The model

used a z-score normalization for features selection and was used to detect complex and obfuscated malware. Experimental assessment had a high level of accuracy and scalability in practical applications. However, although the method was deep learning-based, it did not consider a variety of handcrafted feature representations that could enhance interpretable and robust performance on other datasets. To detect Android malware, Andresini et al. [2] built a graph-based deep learning models using the Graph Neural Network (GNN). This approach transformed the sequences of API calls into a dynamic graph of functions and classified the sequences using GCNs. We also added the GNNExplainer method, which is able to explain predictions based on identifying important API connections.

The accuracy of the model was higher than the traditional methods based on sequences. However, the graph-based models have the disadvantage of being less scalable because of changes in computational complexity. Khoa et al. In [3] proposed a framework, APSDroid, which combines the features of permission-intent and API call graph to boost malware detection. Their approach is to use the feature-level fusion techniques: concatenation and attention. It also contained graph optimization techniques to decrease the complication of the graph while preserving the semantic relationships. Experimental results illustrated good robustness against obfuscation attacks and a high detection accuracy. Yet, combining various intricate components leads to extra complexity and computational expense of the model.

Mohammad et al. [4] presented a hybrid metaheuristic feature selection algorithm by combining Firefly and Binary Bat algorithms, which are proposed to generate more enhanced solutions for optimal subsets of features. This makes the dimensionality of the features to be reduced from thousands to a subset that is computationally more feasible, and on the other hand, fits into the set of most discriminative features. The classifiers used for the classification were ML classifiers like Random Forest and SVM.

Experimental results were better than that of other feature selection techniques. Nonetheless, this technique is highly dependent on feature engineering and might not capture deep semantic patterns. Wang et al. [5] introduced a solution based on classifier fusion for malware detection along with infinite feature selection to solve feature redundancy problems. The algorithm built a weighted graph representing the relationship among features and then selected optimal feature subsets from the resulting subgraphs as features. Multiple classifiers were fused together with a ranking-based fusion strategy to improve detection performance. High F1-scores were obtained in most of the datasets using the method. The ease of framework is complicated, however, due to feature selection and fusion. Yao et al. [6] proposed an adaptive channel attention mechanism and CNN to detect malicious applications through their grayscale image of the APK files. Also, in this model,

knowledge distillation was employed for better generalization and computation. Smaller models achieved high accuracy based on empirical evidence on multiple datasets. Based on feature engineering techniques, Alhogail et al. [7] suggested a two-stage ML-based framework. To enhance the performance, feature reduction methods like chi-square and model-based feature selection were used in the approach. On both the binary and multi-class classifiers, type classification tasks, Random Forest classifiers gave very good results. Makkar et al. In [8], we discussed in detail the ML, DL approaches used to detect Android malicious apps.

Alamro et al. [9] have suggested an ensemble learning-based malware detection model which combines several classifiers and uses optimization techniques to tune the parameters. The algorithm employed by the method included LS-SVM, kernel extreme learning machine, and neural networks to increase the accuracy of classification. The experimental findings showed better performance than the individual models. Nevertheless, the method involves the delicate fine-tuning of parameters and can raise the level of computational complexity.

Museeb et al. [10] designed an Android malware detection model using the Rand Forest classifiers. The method used the presence of the API calls and permissions, as well as feature selection and principal component analysis as dimensionality reduction methods. The model was highly accurate, as well as being computationally efficient. But the use of fixed characteristics can restrict its usefulness in resisting sophisticated obfuscation strategies. Dong et al. [11] proposed a DL-based zero-day malware detection approach with image representations and a zero-shot learning technique. Their model showed high generalization ability to detect malware variants that are not visible.

The AndroDex dataset, introduced by Aurangzeb et al. [12], transforms APK files into image representations to support malware detection using deep learning. Their paper brought out the power of image-based analysis to deal with obfuscation. Tanveer et al. [13] suggested a hybrid DL model that integrates LSTM and GRU network to identify sequential patterns in malware behavior. Alhussen et al. [14] used optimized LSTM and neural network models to detect malware and found the high accuracy of the models due to effective preprocessing and hyperparameter optimization.

Almahmoud et al. [15] suggested an RNN model based on the use of static characteristics of malware, including permission and API calls. Their strategy showed good results in comparison to the conventional techniques. Huang et al. [16] proposed a contrastive learning-based DL method to detect malware based on classes.dex analysis. Their approach enhanced the feature representation and delivered competitive F1-scores.

Moran et al. [17] used dimensionality reduction methods in ML models to analyze the DREBIN dataset. These findings revealed that Random Forest had a high detection rate and less feature space. Ranadheer Kumar et al. [18] examined the use of feature extraction techniques in and demonstrated the enhanced performance of these methods in the Information Gain and Fisher's Score. The approach proposed by Li et al. [19] involves a multimodal feature fusion method that combines the source code and binary representations using pretrained models and attention mechanisms. Their technique enhanced the detection performance of the complementary information. Wang et al. [20] proposed a semantic extraction system is hierarchical feature fusion through deep neural networks. This method was effective in extracting complicated relationships in malware data.

Prasad et al. [21] suggested a permission-based incremental learning model to detect malware and demonstrated high scalability and resistant to adversarial attacks. The permission-based neural network model proposed by Banik et al. [22], based on frequent pattern mining techniques, demonstrated better detection performance on benchmark data. Xu et al. [23] proposed an ensemble-based deep learning model that integrates CNN and DNN models to detect malware, with improved accuracy and reduced computational complexity.

Zhang et al. [24] suggested a two-stage malware detection framework based on feature fusion and cascade deep forest models, which have shown effective performance in detecting malware traffic encrypted with secret keys. Although major progress has been made, existing studies have mainly focused on deep learning models or conventional ML techniques using specific feature representations. Few research efforts have simultaneously explored hybrid feature representations along with ensemble learning approaches. Moreover, fused features are high-dimensional and enhancing the classification accuracy is still a challenge. The proposed work fills these gaps by using DEX image representation along with complementary handcrafted and deep features, dimensionality reduction using the PCA algorithm and testing various ML and ensemble classifiers. The integrated approach offers a coordinated and systematic way to make use of the complementary representations of features.

3. Research Methodology

Figure 1 depicts the entire framework of the proposed DEX-Based Android Malware Detection. The work is initiated by collecting Android APK files from both public accessible malware repositories and sources of benign applications. Each APK is decompiled, and dex files are extracted. This DEX file is important since it contains the main logic of the app. Next, the decoded DEX file is then converted to create an image. With this transformation, the byte patterns of malware and benign apps can be directly examined.

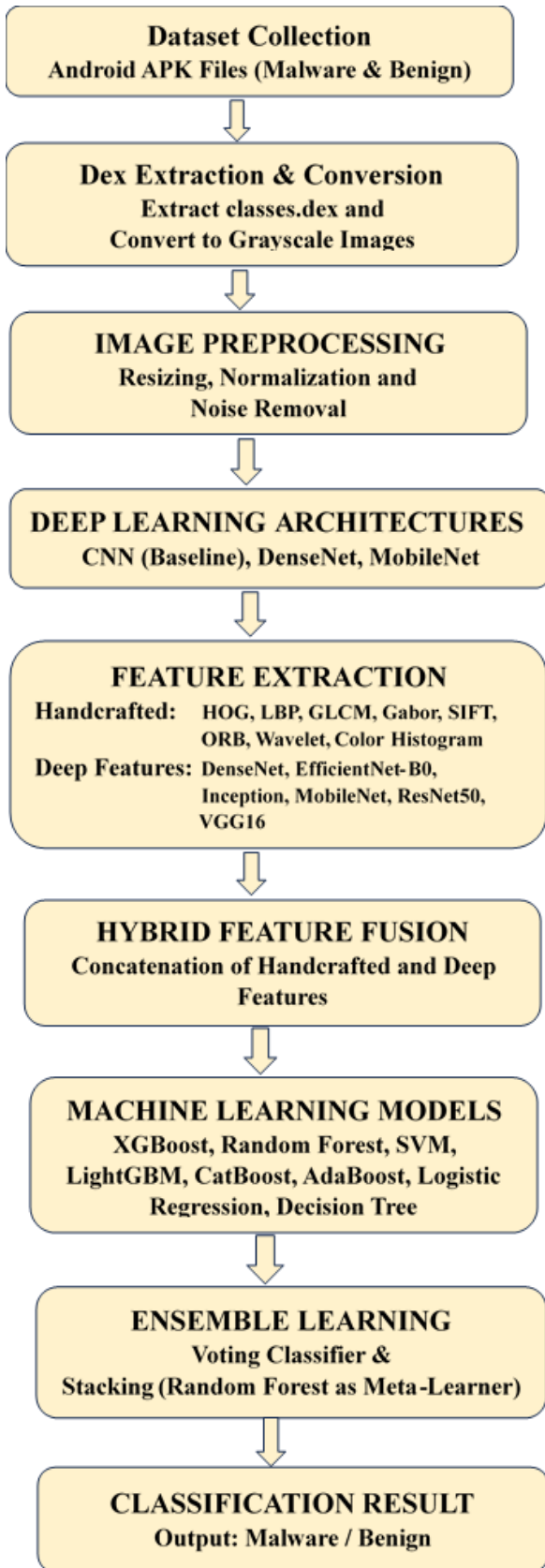


Fig. 1 Proposed framework

After image generation, an initial pre-processing step is performed in order to fix the input size and normalize pixel intensities. The preprocessed DEX images are first fed into a CNN model that acts as the baseline deep-learning classifier. CNN predictions give a preliminary analysis on the discriminative power of DEX images and a bottom line for next phase. In the second step, to improve the performance of classification, some of Transfer Learning (TL) models, such as DenseNet121 and MobileNetV2, are directly applied to DEX images. These architectures take advantage of their deep hierarchical feature extraction to classify the malware and non-malware samples better than the CNN baseline.

In the next phase, the TL models are reused as deep feature extractors but not classifiers. The last global average pooling layers of these models output compact, high-level features that incorporate structural, semantic and spatial information from the DEX images. Simultaneously, a complementary pool of hand-crafted image descriptors (HOG, LBP, SIFT, ORB and GLCM) is computed to account gradient distributions, texture patterns and variance due to keypoints which may not be well captured using deep features. These two feature groups, namely deep embeddings and handcrafted descriptors, are combined with hybrid of feature fusion, wherein all the feature vectors are concatenated to a single representation. The merged feature matrix combines the high-level semantic patterns and low-level texture and keypoint features, providing a rich and detailed representation for each DEX image. Lastly, the concatenated features are fed into several ML classifiers. These models fit decision boundaries on the concatenated feature space and produce a final prediction as malware or benign. This hybrid fusion method provides a substantial improvement of detection accuracy, as shown in the experiments on DEX-image data.

3.1. Dataset Information

In this study, a collection of malware and benign Android applications was created to test the suggested classification system. The samples of malware were taken out of the publicly available VirusShare repository, which contains a big collection of real-world malicious APK files. Trusted sources were used to obtain benign applications (including the Google Play Store and APKPure) to ensure that the samples represent legitimate and non-malicious software.

First, a balanced number of malware and benign APK files were gathered to ensure a balance in the datasets. The preprocessing and conversion stages also involved a minor number of samples being discarded either as a result of corruption or incompatibility. The resultant dataset after the filtering process included 1,488 malware samples and 1,499 benign samples.

The APK files were further processed such that the classes.dex file was extracted, which was in turn converted into grayscale image representations of the file to be analyzed.

In this work, these images acted as the input in both deep learning models and feature extraction techniques applied in this work. The dataset was divided into 90% training and 10% testing sets.

3.2. Preprocessing

Once the data was collected, the APK files were processed to get a unified and structured representation that can be analyzed. Firstly, for each APK file, the classes. The dex file containing compiled application bytecode was extracted. The values of the bytes were then used to look up the intensity of the pixels to create these DEX files as grayscale images. This is to enable the use of image-based analysis techniques for malware detection. The images obtained were then preprocessed in a series of steps to ensure that all images were consistent with regard to the data set. The size of all images was brought to a fixed size to give the deep learning models the same input. The values of the pixel intensities were scaled to a standard range to enable effective and stable model training. Also, simple noise processing was done by deleting invalid or corrupted samples during the conversion process. These preprocessing measures made sure that the dataset was clean, consistent and ready to be used in further steps such as deep learning-based modeling and feature extraction.

3.3. Feature Extraction

To accomplish the task of capturing the various traits of Android malware, a set of combination of handcrafted and deep learning-based features were mined out of the preprocessed grayscale image. The combination of various types of features helps the model to absorb complementary features such as texture, structural patterns, statistical variations, and high-level semantic representations. All the feature extraction techniques employed in this study will be explained below.

3.3.1. Handcrafted Image Feature Extraction

HOG feature descriptor is used to describe the distribution of edge directions in an image. It calculates gradients and pools them into orientation bins, thus capturing the local shape and structural patterns. HOG can be applied in the detection of distinctive edge patterns and structural variations in the visualization of the benign and malicious samples, as far as malware images are concerned.

LBP is a texture describer, which encodes how a pixel relates to the pixels around it. It is used to encode local regions with binary patterns, which is a compact representation of texture information. LBP is especially useful in identifying fine-grained texture variations in malware images, which can reveal subtle variations in code structure of benign and malicious applications. GLCM is a statistical technique that is utilized to examine the spatial dependence among pixel intensities. It derives characteristics like contrast, correlation, energy and homogeneity. These characteristics give

information on the texture and spatial arrangement of the pixel values in the image. GLCM finds repeating patterns and structural dependencies that are predictive of malicious behavior in malware detection.

Gabor filters are used to analyze textures at different orientations and scales. They are very good in extracting frequency and orientation information of images. In this work, Gabor features are used to identify directional patterns and frequency components of malware images, which help to enhance discrimination between different classes.

SIFT is a feature extraction method based on keypoints, which are distinctive points in an image and which are scale and rotation-invariant. It compares local features around these keypoints, which causes it to be resistant to changes in transformations of the image. SIFT can be applied to extract distinctive structural features in malware images that can be reliably matched and classified based on features.

Another effective substitute of SIFT is Oriented ORB, which is a combination of FAST keypoint detection and BRIEF descriptors, which are oriented. It offers high speed and strong feature extraction and is also rotation invariant. ORB is especially effective with large-scale datasets because it is computationally efficient and can assist in identifying key structural features in malware images.

A color histogram is used to represent the distribution of color values in the image. It provides statistical information about the distribution of pixel intensities across the color channels and helps capture differences between benign and malicious samples.

Wavelet transform is applied to break the image into various frequency contents in various scales. This enables both spatial and frequency-domain information to be extracted. The effectiveness of wavelet features is their ability to preserve coarse and fine details in malware images and, consequently, enhance the strength of the feature representation.

3.3.2. Deep Feature Extraction

TL models are pre-trained with large-scale image data and can learn rich hierarchical representations about images. DenseNet provides dense connections, which encourage feature re-use, MobileNet features light and efficient feature extraction, EfficientNet-B0 balances the depth and width of the network, ResNet50 represents deep residual features, Inception represents multi-scale features, and VGG16 represents stable deep feature representations.

The features extracted from these models include high level semantic and structural information found in malware images. The use of several deep feature representations provides the advantage of using different learning

perspectives and enhances the discriminative power of the set of features employed in the classification process.

3.4. Feature Fusion

To enhance the discriminative capability of the classification model, a hybrid feature fusion approach was employed by combining both handcrafted and deep learning-based features. Individually, handcrafted features capture low-level texture, statistical, and structural characteristics, while deep features extracted from pretrained models represent high-level semantic information. However, relying on a single type of feature may not fully represent the complex patterns present in malware images. Therefore, integrating both feature types provides a more comprehensive representation.

In this work, deep features extracted using pretrained models, including DenseNet, MobileNet, EfficientNet-B0, ResNet50, Inception and VGG16, were combined with features that were extracted using multiple handcrafted methods, including HOG, LBP, GLCM, Gabor, SIFT, ORB, histograms, and wavelet transforms. The representation of each feature set was first expressed as a numerical vector that was proportional to the input image.

Prior to fusion, all feature vectors were aligned and normalized so that they could be compatible across different feature scales and distributions. The fusion process was done in a feature-level concatenation strategy, whereby the individual feature vectors were concatenated to form one high-dimensional feature vector. The method maintains the information of all sources of features without losing detail.

The hybrid feature vector resulting is a combination of low-level and high-level information that allows the model to be able to capture a wide range of characteristics of malware images. Such a rich representation contributes to the ability of ML classifiers to distinguish benign from malicious samples, leading to improved classification performance.

The first feature scaling method used was standardization to normalize the distribution of features due to high dimensionality of the fused feature vector. Then, we applied the PCA to reduce dimensions yet retaining 95 % of the variance. This will help in reducing the duplicate information, reducing the complexity of computations and providing efficiency for an ML model.

3.5. Classification Models

The resulting hybrid feature vectors of the fusion process were utilized to train various ML classifiers in detection of malware. The variety of different classification algorithms allows performing an overall assessment and determining the best model to use on the given feature representation.

A series of popular supervised learning algorithms were used, including RFC, SVM, Logistic Regression (LR),

XGBoost, AdaBoost, GBC, LightGBM, CatBoost, and Decision Tree. All the classifiers have different merits as far as learning ability, computational ability and generalization ability are concerned.

RFC is an ensemble-based model, which builds many decision trees and combines their predictions, granting it resistance against overfitting. The Support Vector Machine is efficient with high-dimensional data, and it determines the best decision boundaries to classify data. Logistic Regression is an easy but effective baseline model that is used in binary classification tasks.

Boosting-based algorithms like XGBoost, AdaBoost and Gradient Boosting have an iterative nature in the way they improve model performance by focusing on misclassified samples. Among them, the XGBoost is regarded as efficient and having superior predictive performance. LightGBM is a gradient boosting model which employs histogram-based learning to achieve faster computation and scalability. CatBoost is especially robust when dealing with non-standard distributions of data, and it is also stable when used on a variety of different datasets. Decision Tree is a model that is simple to interpret and is a model that subdivides the feature space in accordance with decision rules.

To achieve a very high level of comparison, the study conducts a comparison among a number of classifiers on the hybrid feature representation to identify the most effective models that can be used to effectively utilize the hybrid feature representation to achieve high-quality malware classification.

3.6. Ensemble Learning

To further enhance classification performance and improve the generalization capability of the proposed framework, ensemble learning methods were utilized by averaging the predictions of multiple ML models. Ensemble methods can be used to minimize model variance and bias by exploiting the strengths of individual classifiers and, therefore, make more robust and accurate predictions.

The two ensemble strategies that were used in this paper included voting and stacking classifiers. The voting classifier is a technique that takes the results of several base models and averages the results of these models. In particular, a weak voting scheme was used, where the final prediction is the average of the class probabilities of the chosen classifiers. In this work, Support Vector Machine (SVM), XGBoost and LightGBM were selected as base learners in the voting ensemble since they perform well individually.

Besides voting, a stacking classifier was also developed in order to enhance predictive performance even more. In stacking, the meta-learner is trained on the predictions of the base models, allowing the system to learn which combinations of predictions best predict the outcome. The use of SVM,

XGBoost, and LightGBM as base classifiers and the use of the Random Forest as the meta-learner in this approach have been used. The meta-learner was trained to integrate the forecasts of the underlying models and generate the ultimate classification output.

Ensemble learning is used to enable the model to learn complementary decision patterns among the various classifiers, resulting in a higher accuracy and stability. This method can play a crucial role in improving the performance of the proposed system as opposed to individual ML models.

4. Results and Discussion

The proposed malware classification framework was deployed and tested on Colab, which offers a cloud-based platform with GPU support to effectively train and experiment with the model. The experiments have been performed on the pre-processed data.

To thoroughly evaluate the performance of the suggested approach, several models have been tested at various steps, including a baseline CNN, transfer learning models, ML classifiers that are trained on hybrid features and ensemble learning methods.

The results of every model were examined in terms of conventional evaluation tools and compared with each other to find out which approach works the best.

The findings are well arranged in a systematic way in order to emphasize the role of each step of the proposed framework. First, we discuss the performance of the CNN model as a baseline. This is then followed by the analysis of transfer learning models, using which pretrained architectures are used to enhance the extraction of features.

After that, the results of ML classifiers on the combined feature set are discussed. Lastly, ensemble learning techniques, such as voting and stacking classifiers, are also examined to show that they are effective in improving classification accuracy.

The CNN was trained for 15 epochs using the Adam optimizer with binary cross-entropy loss. DenseNet121 and MobileNetV2 used ImageNet-pretrained weights with input images of size $128 \times 128 \times 3$, and their convolutional base layers were frozen during training. DenseNet121 was trained for 10 epochs with Adam (learning rate 0.00005), while MobileNetV2 was trained for 10 epochs with Adam (learning rate 0.0001).

4.1. CNN Results

The baseline model was a CNN to assess how effectively the malware can be classified through image. The model structure is based on a series of convolutional layers that are

followed by max-pooling layers and fully connected layers used to classify binary. The training of the network was done with the preprocessed grayscale image dataset.

Epoch-wise accuracy and loss measures were used to evaluate the performance of the CNN model. As indicated by Figures 2 and 3, the accuracy of the training steadily increases with the number of epochs, which indicates that the feature representations are learning effectively. Some fluctuations in the validation accuracy between epochs were observed; hence, the average validation accuracy was taken into consideration to evaluate the performance, which was found to be about 82%. The loss curves show a stable reduction in both the training and validation loss, which implies a stable convergence of the model. Nevertheless, lack of information in the difference between training and validation performance indicates moderate generalization ability. On the whole, the CNN model offers a stable baseline to compare with more sophisticated models, such as transfer learning, hybrid feature-based models.

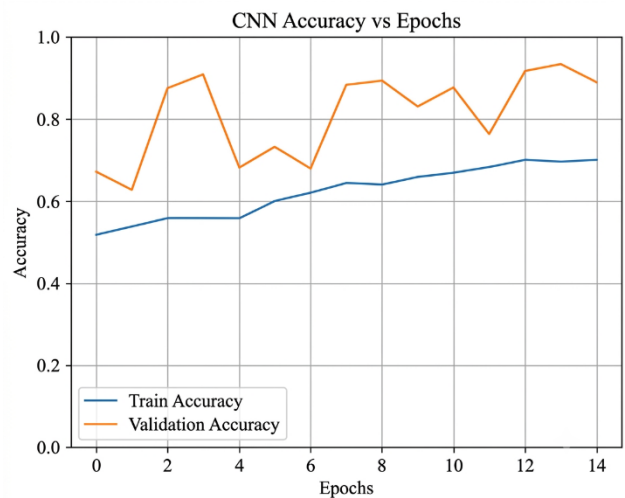


Fig. 2 Epoch wise accuracy with CNN

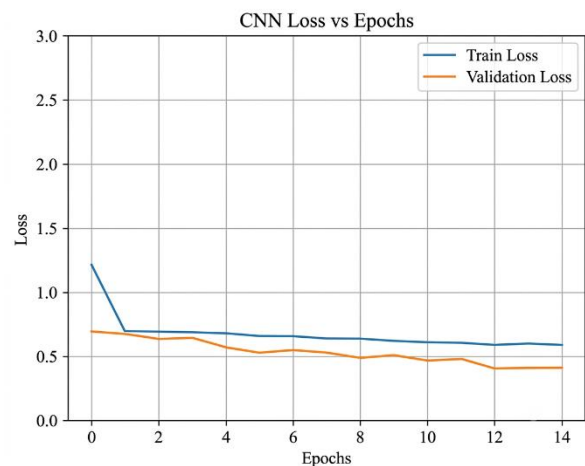


Fig. 3 Epoch wise loss with CNN

The CNN model suggests that the spatial and structural patterns of DEX images are discriminative to be utilized as a malware identification feature without any extra feature extraction or transfer learning. Figures 2 and Figure 3 shows epoch wise accuracy and loss with the CNN model.

4.2. Transfer Learning Results

To enhance the performance as compared to the baseline CNN model, transfer learning methods were used with the use of pretrained architectures, DenseNet and MobileNet. The models make use of pretrained weights in order to find more discriminative features in the malware images. The DenseNet model results are shown in Figure 4 and Figure 5. As it can be seen, the training accuracy steadily increases with every epoch, whereas the validation accuracy grows quickly and reaches its peak (89%) at the end of the epoch. The curves of loss show that the training and validation loss decrease consistently, showing steady learning and successful convergence. Overall, the DenseNet model had a total accuracy of about 89%, which is higher than the baseline CNN model.

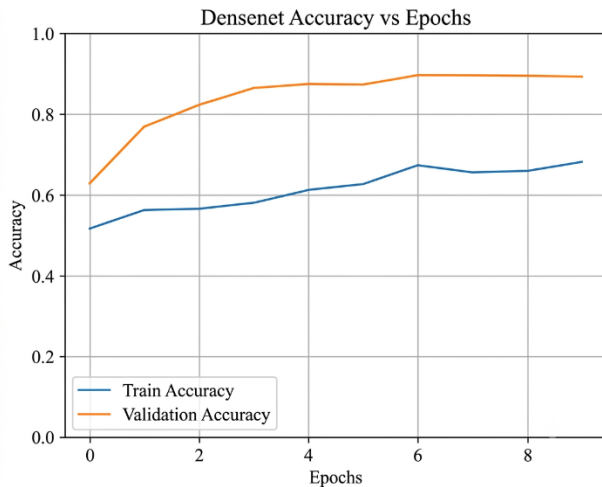


Fig. 4 Accuracy with DenseNet

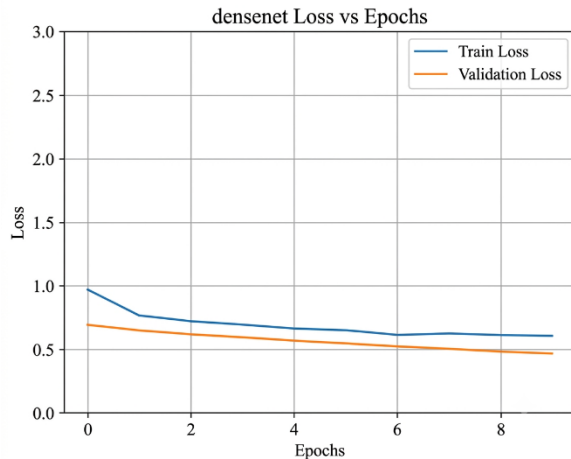


Fig. 5 Loss with DenseNet

The results of the MobileNet model are shown in Figure 6 and Figure 7. The validation accuracy gradually increases and reaches the stabilization level of 85, which corresponds to the fact that the feature extraction ability is effective.

The training accuracy is quite constant, whereas training and validation loss are decreasing in trend, which proves correct convergence. MobileNet is slightly less accurate than DenseNet but has computationally efficiency because of its lightweight design.

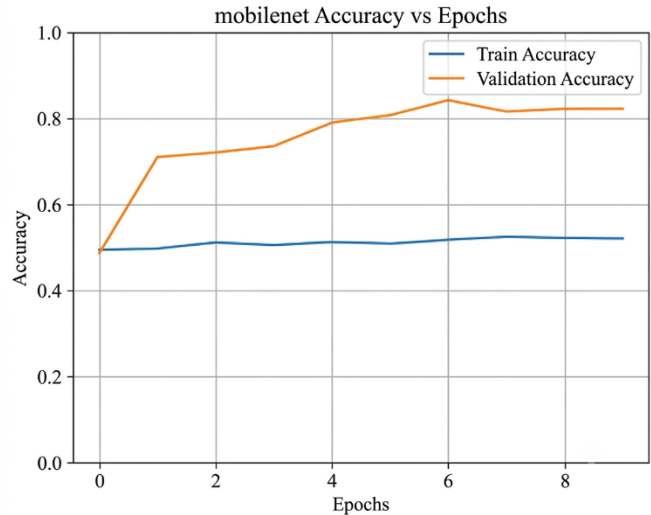


Fig. 6 Accuracy with MobileNet

On the whole, the transfer learning models perform considerably better than the baseline CNN as a result of using pretrained feature representations. Of the models tested, DenseNet performed best, followed by MobileNet. These findings show that transfer learning is effective when used in improving the accuracy of malware classification.

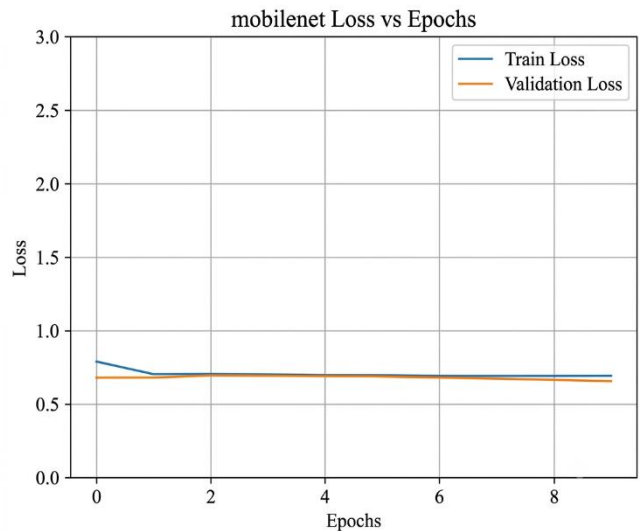


Fig. 7 Loss with MobileNet

4.3. Machine Learning Results

The obtained hybrid feature vectors of feature fusion were trained to run various ML classifiers to classify malware. A wide range of supervised learning algorithms was taken into consideration to identify the efficiency of the feature representation. The fused feature set was trained and tested on these models to examine their ability to differentiate between benign and malicious samples.

Table 1. Performance with ML models and fused features

Model	Precision	Recall	F1	Accuracy
RFC	0.86	0.84	0.84	84.62
SVM	0.90	0.90	0.90	89.97
Log. Reg	0.87	0.86	0.87	86.62
XGB	0.91	0.91	0.91	91.30
AdaBoost	0.88	0.88	0.88	87.96
GBC	0.88	0.88	0.88	88.29
DTC	0.79	0.79	0.79	78.93
LightGBM	0.90	0.90	0.90	90.30
CatBoost	0.88	0.88	0.88	87.63

Because the fused feature space is of high dimensionality, feature scaling was initially used, with standardization to normalize the distributions of features. PCA was then used to decrease the number of dimensions, and at the same time, 95%

of the total variance would be retained. The dimensionality of the feature set was reduced to 1,554 features out of 29,307. This transformation served to enhance the efficiency of the models and also to reduce redundancy in the feature space.

Table 1, Figure 8 and Figure 9 present the performance of the different ML classifiers on the PCA-transformed hybrid feature set. The findings indicate that the majority of the models obtained balanced performance values, indicating stable classification performance across the two classes (benign and malware).

XGBoost had the highest accuracy of 91.30%, as well as the best overall accuracy, recall, and F1-score of 0.91. LightGBM and Support Vector Machine also reflected good performances with accuracy of 90.30% and 89.97%, respectively, and high accuracy and recall rates at all times, which is to the credit of their effectiveness in addressing high-dimensional feature representations.

The boosting-based models, such as AdaBoost and GBC, showed competitive performance with F1-scores of about 0.88, which depicts a sound classification capability. The results of the Logistic Regression were also consistent with equal precision and recall, with an accuracy of 86.62%.

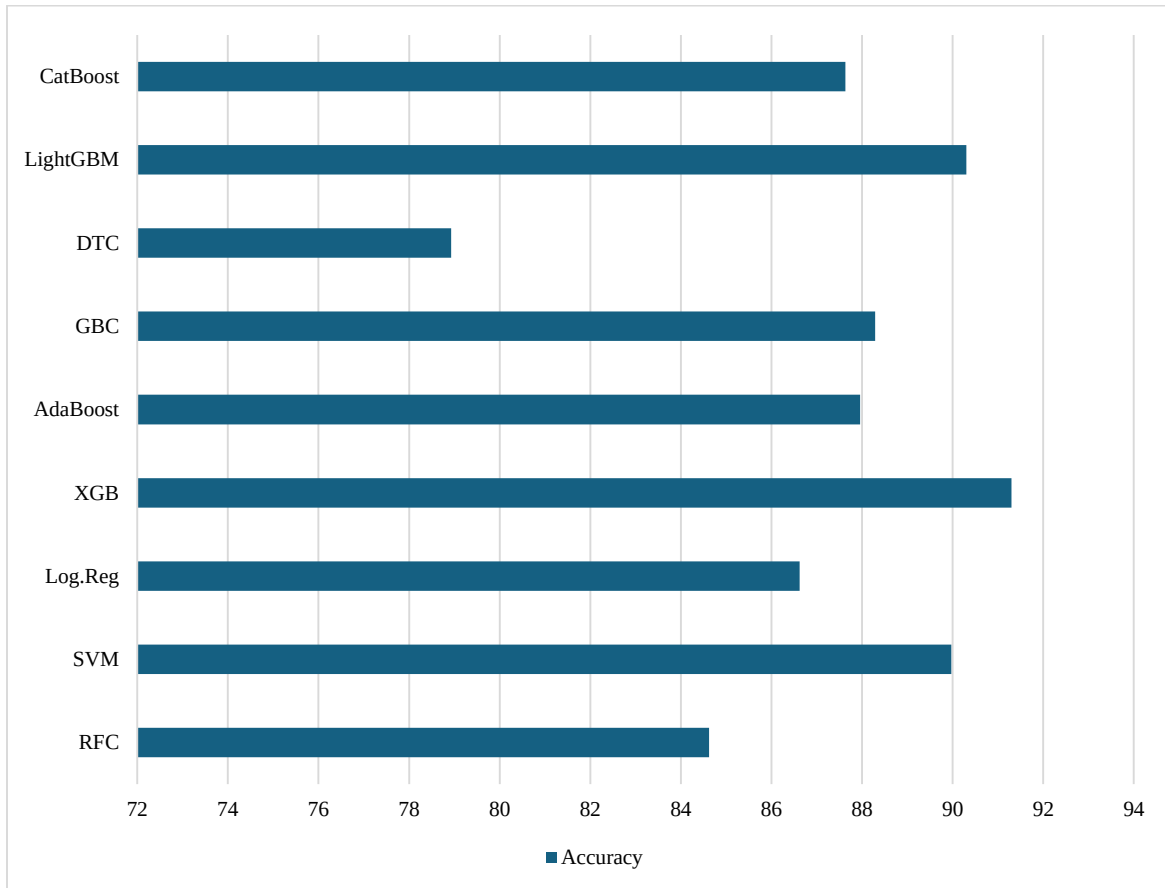


Fig. 8 Accuracy with ML models

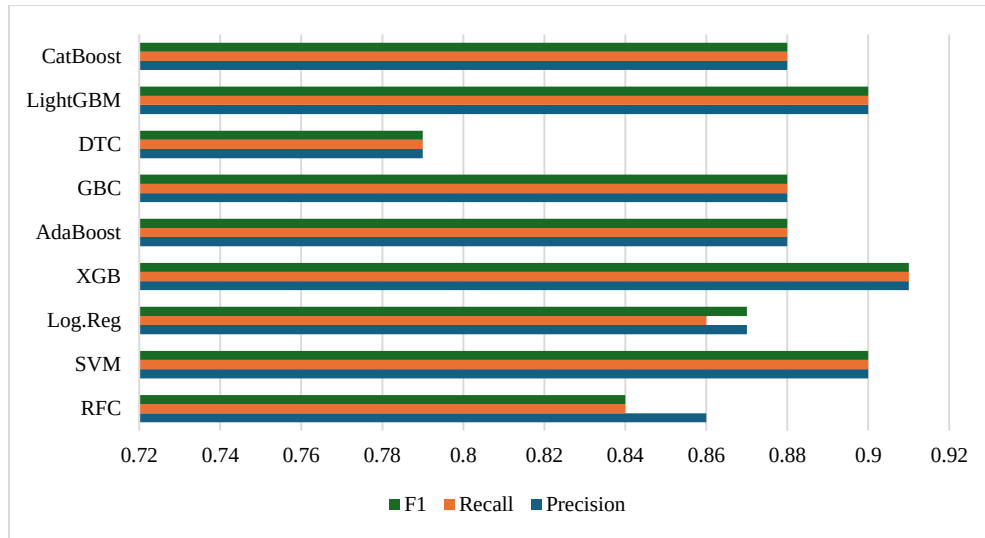


Fig. 9 Precision, Recall, F1 score with ML models

Conversely, the recall of malware samples in random forest was relatively low, and this influenced the overall performance. Decision Tree recorded the lowest accuracy at 78.93 with relatively lower precision and recall values, which implies that it is not as effective as decision trees in modelling the complex interaction of features.

4.4. Results with Ensemble Models and Fused Features

These findings show that boosting-based and kernel-based models work better with the suggested hybrid feature space. The uniformity of the results also help to prove the strength of the suggested approach. To further enhance classification performance, the ensemble learning methods were used by integrating the outputs of several ML models. This paper compared two methods of ensembles: voting and stacking classifiers. Both approaches used SVM, XGBoost and LightGBM as base learners because they have good individual performance.

Table 2 and Figure 10 give the performance of the ensemble models. Voting classifier given precision, recall, F1 and accuracy values of 0.91, 0.91, 0.91, and 90.97. This means that prediction made by a number of models increases the stability of classification and minimizes the bias of an individual model.

Table 2. Performance with ensemble models

Model	Precision	Recall	F1	Accuracy
Voting	0.91	0.91	0.91	90.97
Stacking (RF)	0.92	0.91	0.92	91.64

To further improve performance, a stacking classifier was adopted with a meta-learner of Random Forest. The stacking model was the most accurate with at 91.64%, beating the performance of both individual ML models and the voting ensemble. It further obtained a higher precision and F1-score values reflecting a high overall classification ability.

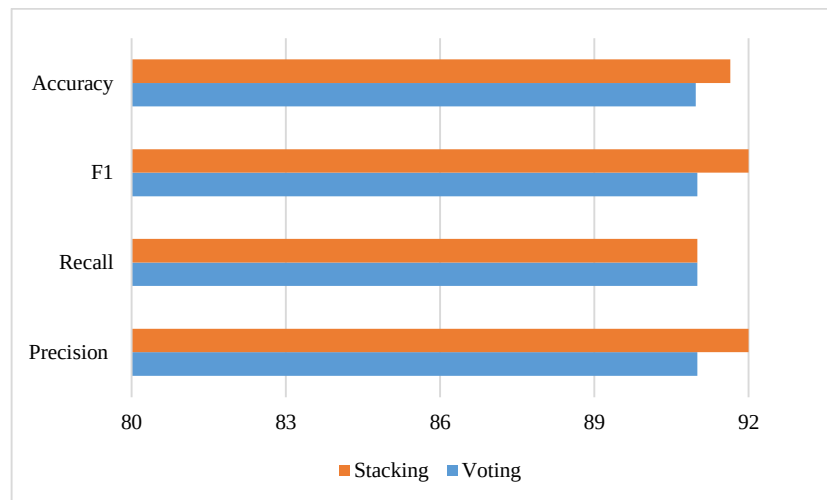


Fig. 10 Performance with ensemble models

The better performance of the stacking model is explained by the possibility to learn the best combinations of predictions of various base classifiers. With the help of the advantages of SVM, XGBoost, and LightGBM, the meta-learner is capable of maximizing the boundaries of the decision-making and minimizing errors in the classification.

Figure 11 shows the ROC curve of the stacking model. The model had an AUC of about 0.95, which showed the model has excellent discriminative power between malware and benign samples. This ROC curve is another validation of the strength and consistency of the projected ensemble approach.

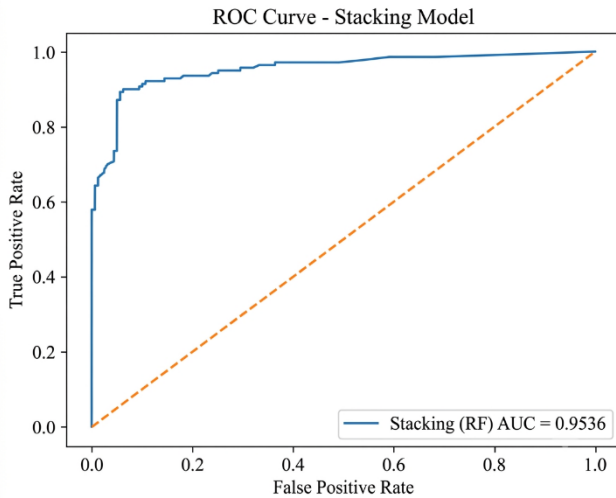


Fig. 11 ROC curve with stacking model

Overall, the ensemble learning strategies improved classification performance, with the stacking classifier providing the best results among all evaluated models. This validates the efficiency of combining multiple classifiers for robust malware detection.

The enhanced performance of the framework can be accredited to the complementary information provided by the fused feature representation and the combination of different classifiers. The handcrafted features capture texture, structural, statistical, and local patterns from the DEX images, while the deep features provide higher-level representations. PCA further reduces the high-dimensional fused feature space while retaining 95% of the variance. Among the evaluated classifiers, XGBoost achieved 91.30% accuracy, while the stacking model achieved 91.64% by combining the predictions of SVM, XGBoost, and LightGBM through Random Forest as the meta-learner. This combination allows

the stacking model to exploit complementary decision patterns from the base classifiers, resulting in improved performance compared with the individual models and voting ensemble.

4.5. Comparison with Existing Works

Table 3. Comparison with previous works

Model	Accuracy
XGBoost [12]	88%
CNN+RNN [25]	89.14%
Stacking (RF)	91.64%

In Table 3, the proposed method is compared with existing methods. The accuracy is 88% and 89.14% for classifiers XGBoost [12] and CNN+RNN [25]. The proposed Stacking (RF) model recorded an accuracy of 91.64%, which is much better than the classification performance of existing selected approaches.

5. Conclusion

This work presented a framework for Android malware classification by integrating deep learning, handcrafted feature extraction and ensemble learning techniques. First, a baseline CNN model was created, and then transfer learning techniques were employed to improve the feature representation of the CNN with pre-trained models. Several handcrafted and deep features were extracted and combined to create a hybrid feature set for further boost of the performance. The features were standardized and reduced using PCA to remove any redundant features and to make computation efficient. Several classifiers of ML were then tested on the transformed feature space. Hereof, boosting-based models performed best, with XGBoost performing the best of all the single classifiers. Several ensemble learning techniques were used to further improve classification. The stacking classifier with Random Forest as meta-learner had the highest overall accuracy of 91.64%, and a high ROC-AUC score, which meant it had a strong discriminative capability. The results show that the proposed hybrid feature-based ensemble approach gives improvement over the single deep learning models. Overall, this work highlights the effectiveness of combining handcrafted features, deep learning representations, and ensemble techniques for robust malware detection. Future research could involve investigating more sophisticated feature selection algorithms, deployment in real-time and the expansion of the algorithm to accommodate a wider range of and larger malware datasets.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

References

- [1] Anuradha Anuradha, Arun Singh Chouhan, and S. Srinivas Rao, "Improving Malware Detection Performance using Hybrid Deep Representation Learning with Heuristic Search Algorithms," *Scientific Reports*, vol. 16, no. 1, pp. 1-30, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]

- [2] Giuseppina Andresini et al., "Anakin: Explainable Android Malware Detection with Graph Neural Networks," *Cybersecurity*, vol. 9, no. 1, pp. 1-37, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [3] Nghi Hoang Khoa et al., "Android Malware Detection by using Graph Optimization of Static Features based on Pre-Trained Language Models," *Information and Software Technology*, vol. 192, pp. 1-19, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [4] Rami M. Mohammad, "Android Malware Detection using a Novel Binary Firefly Bat Feature Selection Algorithm," *Information Security Journal: A Global Perspective*, pp. 1-27, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [5] Shengran Wang et al., "A Novel Android Malware Detection Method based on Cwinfs and MPTACF Optimization," *Journal of Information Security and Applications*, vol. 99, pp. 1-13, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [6] Liangwei Yao et al., "A Lightweight Android Malware Detection Framework based on Markov Images and Knowledge Distillation," *IEEE Internet of Things Journal*, vol. 13, no. 3, pp. 5224-5241, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [7] Areej Alhogail, and Rawan Abdulaziz Alharbi, "Effective ML-based Android Malware Detection and Categorization," *Electronics*, vol. 14, no. 8, pp. 1-22, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [8] Inderpreet Singh Makkar et al., "Android Malware Detection: Necessity, Applications and Future Direction," *IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation*, Gwalior, India, pp. 1-6, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [9] Hayam Alamro et al., "Automated Android Malware Detection using Optimal Ensemble Learning Approach for Cybersecurity," *IEEE Access*, vol. 11, pp. 72509-72517, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [10] Abdul Museeb et al., "Android Malware Detection using API Calls and Permissions with Random Forest Classifier," *IEEE Access*, vol. 14, pp. 6464-6480, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [11] Shi Dong et al., "Android Zero-Day Guard: Zero-Shot Malware Detection using Deep Learning and Generative Models," *IEEE Transactions on Network and Service Management*, vol. 23, pp. 3251-3267, 2026. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [12] Sana Aurangzeb et al., "AndroDex: Android Dex Images of Obfuscated Malware," *Scientific Data*, vol. 11, no. 1, pp. 1-10, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [13] Muhammad Usama Tanveer et al., "Malware-SeqGuard: An Approach Utilizing LSTM and GRU for Effective Detection of Evolving Malware in Android Environments," *IEEE Access*, vol. 13, pp. 117355-117373, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [14] Ahmed Alhussen, "Advanced Android Malware Detection through Deep Learning Optimization," *Engineering, Technology and Applied Science Research*, vol. 14, no. 3, pp. 14552-14557, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [15] Mothanna Almahmoud, Dalia Alzu'bi, and Qussai Yaseen, "ReDroidDet: Android Malware Detection based on Recurrent Neural Network," *Procedia Computer Science*, vol. 184, pp. 841-846, 2021. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [16] Yuanming Huang et al., "ACE: A Static Android Malware Detection Method based on Supervised Contrastive Learning," *IEEE Internet of Things Journal*, vol. 12, no. 13, pp. 23550-23562, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [17] Pablo Morán et al., "Machine Learning Models and Dimensionality Reduction for Improving the Android Malware Detection," *PeerJ Computer Science*, vol. 10, pp. 1-31, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [18] K.S. Ranadheer Kumar, K.S. Ranadheer Kumar, and Jagadish Gurrula, "Enhancing Android Malware Detection through Filter-based Feature Selection and Machine Learning Classification," *Journal of Electrical Systems*, vol. 20, no. 3, pp. 2801-2809 2024. [[Google Scholar](#)] [[Publisher Link](#)]
- [19] Xun Li et al., "Multimodal Fusion for Android Malware Detection based on Large Pre-Trained Models," *IEEE Transactions on Software Engineering*, vol. 51, no. 5, pp. 1569-1590, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [20] Tianbo Wang et al., "ArchSentry: Enhanced Android Malware Detection via Hierarchical Semantic Extraction," *IEEE Transactions on Network and Service Management*, vol. 22, no. 3, pp. 2822-2837, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [21] Arvind Prasad et al., "PermGuard: A Scalable Framework for Android Malware Detection using Permission-to-Exploitation Mapping," *IEEE Access*, vol. 13, pp. 507-528, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [22] Abhinandan Banik, and Jyoti Prakash Singh, "Android Malware Detection by Correlated Real Permission Couples using FP Growth Algorithm and Neural Networks," *IEEE Access*, vol. 11, pp. 124996-125010, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [23] Xiaolong Xu et al., "DCEL: Classifier Fusion Model for Android Malware Detection," *Journal of Systems Engineering and Electronics*, vol. 35, no. 1, pp. 163-177, 2024. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [24] Xueqin Zhang et al., "Detection of Android Malware based on Deep Forest and Feature Enhancement," *IEEE Access*, vol. 11, pp. 29344-29359, 2023. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]
- [25] Sriram Kotha et al., "Android Malware Detection using Deep Learning," *International Conference on Advancements in Smart, Secure and Intelligent Computing*, Bhubaneswar, India, pp. 1-4, 2025. [[CrossRef](#)] [[Google Scholar](#)] [[Publisher Link](#)]