Future Technology
Open Access Journal
ISSN 2832-0379

May 2026| Volume 05 | Issue 02 | Pages 129-137

Journal homepage: https://fupubco.com/futech

https://doi.org/10.55670/fpll.futech.5.2.12

Future Publishing LLC

Article

A hybrid deep-handcrafted feature fusion framework for image based android malware detection

Kavitha Mudunuru*, M. Usha Rani

Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam, Tirupathi, India

ARTICLE INFO

Article history:
Received 15 October 2025
Received in revised form
21 December 2025
Accepted 23 January 2026

Keywords:
Android malware detection, Transfer learning, Feature fusion, Machine learning, DEX image analysis, Texture descriptors

*Corresponding author
Email address:
kavitha.mudunuru23@gmail.com

DOI: 10.55670/fpll.futech.5.2.12

ABSTRACT

The rapid growth of Android applications has been accompanied by a corresponding rise in sophisticated malware, posing significant challenges to traditional detection mechanisms. Image-based malware analysis has recently emerged as an effective alternative, as transforming executable bytecode into grayscale images reveals structural, spatial and statistical patterns that remain difficult to conceal. Motivated by this, the present study proposes a hybrid learning framework for Android malware detection using grayscale images generated exclusively from DEX bytecode segments. Multiple deep feature extractors based on Transfer Learning architectures—including DenseNet121, MobileNetV2 and InceptionV3 are employed to obtain high-level semantic representations from DEX images, while handcrafted descriptors such as HOG, SIFT, ORB, LBP and GLCM capture complementary gradient and texture characteristics. The fused feature representations are evaluated using several machine learning classifiers, including Random Forest, Logistic Regression, SVM, KNN and Naïve Bayes. Experimental results demonstrate that the DEX image representation yields highly discriminative patterns, achieving a maximum accuracy of 94.40% with Random Forest and 94.33% with Logistic Regression. These findings confirm the effectiveness of DEX-driven image analysis and hybrid feature fusion as a robust, scalable solution for Android malware detection.

1. Introduction

Android is now the most popular mobile operating system in the world and, therefore, an attractive target for broad-based malware attacks. The platform's open design, popularity of third-party app stores and lazy re-packaging of legitimate apps have also all been blamed for the explosive growth in malicious Android applications. Variants of Modern Malware use sophisticated evasion techniques such as code obfuscation, control-flow manipulation, reflection, packing and deploy dynamic payload. These methods heavily undermine classical malware detectors that largely depend on static signature, opcode sequences or heuristic patterns and can be easily tampered by attackers. Therefore, there is an increasing demand for intelligent detection mechanisms which are robust against the legal obfuscation. Among the next-step research directions, an image-based malware detector is becoming a new research topic since raw bytecode can be abstracted as a visual pattern. Upon transforming a virtual byte stream into a grayscale image representation of the applicable component, an output reveals structural and statistical properties about the code beneath. These types of images frequently disclose reoccurring textures, doing-space structure, block arrangement on implementation and changes in entropy that distinguish benign from malicious applications. Significantly, these visual fingerprints are at least partially preserved across heavy code obfuscation and hence they potentially provide the promising avenue for identifying morphing or polymorphic malware families. In Android apps, the DEX file (i.e., compiled Dalvik bytecode) is especially interesting for image-based analysis, as it directly encodes the machine instructions that control an app's behavior. Motivated by these findings, this work concentrated on DEX-image–based Android malware detection. The goal is to verify whether DEX bytecode visual patterns are suitable for generating highly discriminative feature representations for ML classifiers. In order to accomplish this, the study evaluates state of the art deep semantic features and handcrafted texture descriptors derived from DEX images and investigated how efficient these are when integrated together into a single feature representation. This work differs from previous Android malware detection works by proposing a uniform DEX-image–driven hybrid feature fusion framework based on deep semantic representations and handcrafted image

descriptors in a consistent experimental pipeline. In contrast to previous studies that built on an exclusive use of either deep learning models or handcrafted features, in this study we methodically compare CNN-based classification including several Transfer Learning architectures and machine learning models over the same DEX-image dataset with a standardized preprocessing. The proposed methodology considers complementary features interactions by integrating deep embeddings generated from pretrained CNNs to texture, gradient and key point based handcrafted descriptors, providing better detection performance and satisfactory invariance against code obfuscation.

The objectives of this work can be outlined as follows:

- Designing an effective DEX-image–based malware detection model based on CNN and Transfer Learning for Android applications and evaluate its overall performance.
- To explore the validity of deep feature extracting via pretrained architectures (DenseNet121, MobileNetV2 and InceptionV3) on DEX-derived grayscale images.
- To generate and process handcrafted image descriptors such as HOG, SIFT, ORB, LBP and GLCM in order to capture complementary structural and texture-level patterns from executabled bytecode images.
- To devise a hybrid feature fusion approach that integrates deep semantic embeddings and handcrafted descriptors into a concatenated global feature.
- To test the fused feature space with many machine learning algorithms and benchmark by accuracy, precision, recall and F1-score.

2. Literature survey

Rakibul Hasan et al. [1] conducted a study on Android malware detection based on feature scaling, feature selection and machine learning classifiers. Their work focused on the impact of data preprocessing including normalization and dimension reduction in both stability and performance of classifiers. The effect of the feature transformation on the accuracy in the malware detection was measured by using various learning models. The authors also concluded that a solid preprocessing pipeline is essential to enhance detection confidence over sizeable Android data collections. Amarjyoti Pathak et al. [2] presented a static Android malware detection method by permission inspection. This research concentrated on finding the most discriminative permissions with regard to malicious activities that have a high influence to malwares. By decreasing the feature dimension and removing redundant permissions, the study enhanced computational efficiency of the method under reliable classification rate. Their research emphasized the need to analyze permission levels for lightweight malware prevention systems. Arvind Mahindru et al. [3] proposed a multi-step feature selection approach for Android malware detection that is based on statistical hypothesis testing, regression analysis and correlation. We executed the framework on a rich collection of Android applications, which is called a large-scale dataset. Their method effectively eliminated the redundancy of features and enhanced the generalizability of classifying. It was shown from the study that systematic feature selection improves on robustness and scalability of the malware detectors.

Kaur et al. [4] studied malware detection on Android applications, by means of a mixture of static features such as permissions and API calls, unsupervised clustering and dimensionality reduction methods. The effectiveness of various representations was demonstrated, and several machine learning models were compared. Their result demonstrated that integrating exploratory data analysis with dimensionality reduction enhances adaptability to malware dynamics. The work emphasized the role of feature engineering in static malware analysis. Muhammad Umar Rashid et al. [5] introduced an Android malware detection system based on deep learning that combines permissions, intents and API calls. The model has been created for enhancing the resistance to code obfuscation techniques which are used by malware authors. Experiments on a large variety of malware families showed increased robustness. The work demonstrated the effectiveness of deep learning models with complex android threat behaviors.

Aamir et al. [6] suggested a CNN model for Android malware detection via static features of applications. The work showed that CNNs can learn complex feature representations in a purely data-driven manner without explicit feature engineering. The authors demonstrated that deep learning classifiers are more accurate than conventional ones on the static features as long as there is enough information present. They experimentally verified the feasibility of using CNNs in Android security researches. Sonya et al. [7] proposed a hybrid malware detection approach, by combining static APK analysis with dynamic runtime behavior observation. The system under consideration combines multiple feature sources in an attempt to enhance the malware characterization. Using static and dynamic based evidence, the framework increases the risk of the detection even in case of advanced malware variants. The research stressed on the importance of using multiple sources to detect as many malware as possible. Pathak et al. [8] presented an Android malware detection framework based on permission that ranked the importance of features. Their approach is focusing to find the most influential permissions contributing in such a malicious behavior. With the dimensionality reduction, the model was executed in a faster manner and meanwhile ignored no fewer of useful detection. It was shown in the work that this would lead to enhanced performance at efficiency without compromising classification effectiveness.

Ahmed et al. [9] compared traditional machine learning classifiers for static features in the domain of Android malware. Models such as Naïve bayes, SVM, KNN and Random Forest are experimented. Their study shows that for static Android datasets, ensemble-based and margin-based classifiers work well. The work also offered insights into the classifier choice for malware detection. Haq et al. [10] published an article on Android malware detection with static, dynamic and hybrid method was researched. The authors compared different classes of machine learning models for the effect of AUC in case of feature combinations. Their work showed that combining multiple analysis viewpoints provides more robust protection against various malware behaviors. Christiana et al. [11] presented an in-depth survey machine learning algorithms used for Android malware recognition. The survey further established the drawbacks of signature-based systems and analyzed how learning-based techniques have evolved. The bottlenecks considered for the algorithm were dataset imbalance, obfuscation and scalability. The authors underlined the fact that there is need of more robust and dynamic malwares detection model. Lee et al. [12] investigated GA for feature selection in Android malware detection, with deploying large static feature sets. Their method made features less redundant and preserved the discriminative power. Experimental study presented that the computational efficiency was indeed enhanced by the evolutionary feature

selection. The contribution of the 66 this study lies in showing the potency of optimization-based feature selection reduced techniques. Palma et al. [13] studied the Android malware detection on feature selection and dimension reduction. The authors prioritized interpretability of the model, highlighting the most relevant permissions. Their results showed that well-chosen features can help in the consistency and interpretability of detection. The work campaigned for transparent feature selection in the malware analysis. Chowdhury et al. [14] provided a comprehensive overview of Android malware detection techniques including supervised, unsupervised, and deep learning based methods. The paper examined popular datasets, evaluation metrics and experimental protocols. A number of open problems were identified: the absence of benchmark datasets; spreading use of malware obfuscation. The authors described the future works for Android security.

Odat et al. [15] presented a co-existence–oriented feature extraction technique based on permission, API call pairs.[–Such combinations. The model utilized frequent pattern mining to detect related features. Their technique enhanced malware characterization using relationship among static attributes. The research proved the effectiveness of pattern-based feature modeling. Kumar et al. [16] presented a two-level classification framework for detecting Android malware based on both static and dynamic analysis. The application metadata was also combined with the runtime behavior in order to enhance the reliability of detection. It made them more resistant to code obfuscation and dynamic payload execution. The paper emphasized the significance in behavior of the art in malware classification. Lakshmanarao et al. [17] presented an Android malware detection method by analyzing opcode sequences with the help of RNNs. The architecture exploited temporal dependency between opcode patterns to separate malware from benign applications. Their results validated the usefulness of sequence learning in capturing executable behavior.

Lu et al. [18] proposed a hybrid deep neural model by integrating static and dynamic feature learning. The introduced paradigm was designed to enhance detection strength over obfuscated malware samples. Their approach used both synergistic and complementary learning to improve the classification robustness. The research showed increased agility in the face of changing malware threats. Navaneethan et al. [19] proposed a real-time Android security application that detected malware, malicious URL and SQL injection attacks. It was also used to combine many threat detection modules to a single application. Their research showed the possibility of using smart security systems on mobile devices. The research underscored the applicability of mobile threat detection in practice. AlOmari et al. [20] investigated the effects of enhanced data balancing, normalization and dimensionality reduction for Android malware detection. In this paper several classifier are compared under different preprocessing conditions. Their results indicated the importance of proper preprocessing methods for maintaining classifier stability. They emphasized the necessity of data preparing for malware analysis pipeline.

3. Methodology

Figure 1 depicts the entire framework of 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. Then, the decoded DEX file is transformed to generate a gray image. This transformation makes it possible to directly examine the byte patterns of malware and benign applications. After image generation, an initial pre-processing step is performed in order to fix the input size and normalize pixel intensities. The pre-processed 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 future study. In the next step, to enhance classification performance, some Transfer Learning (TL) models like InceptionV3, DenseNet121 and MobileNetV2 are directly used on 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.

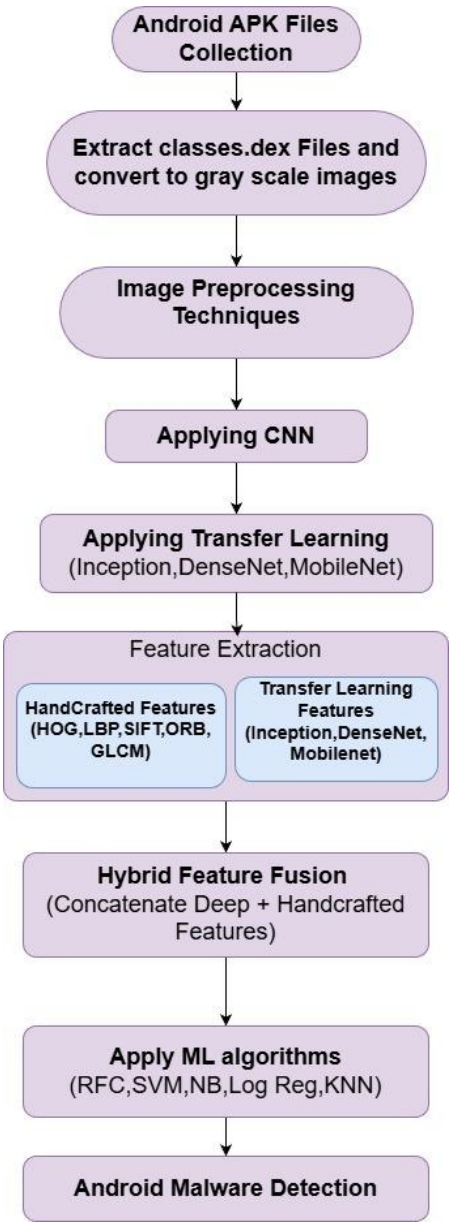

Figure 1. Proposed framework

3.1 Dataset information and preprocessing

The dataset used in this work is based on the Android application corpus originally formulated in [21]. The dataset was a balanced dataset of benign applications downloaded from app store, and malicious samples extracted from popular malware repositories. Several image representations were explored, and found that down sampled grayscale images derived from DEX demonstrate better discriminative power comparing to other possible transformations at APK level. This gain is likely due to the fact that DEX-based grayscale images retain the spatial distribution, instruction density and structural features of executable bytecode. Contrary, DEX images don't include assets or metadata that may be duplicated in app bundles unlike whole-APK transformations. In addition, grayscale encoding is a compact and clean representation for convolutional feature learning and it can well differentiate benign/malicious patterns which are moreover obfuscating. DEX file itself serves as the direct encoding of executable bytecode. Motivated by these observations, the present study exclusively utilizes the DEX-image dataset, consisting of 1987 valid DEX images representing both benign and malicious samples. There are 988 malware images and 999 benign images in the dataset.

For each application, All the images were resized to the same resolution (224 × 224) and normalized (mim-max normalization) between [0, 1] range to have consistency between them with deep learning design. An 80%–20% stratified split of the dataset into training and testing sets was applied to ensure that the ratio of benign and malware classes did not change across splits. A fixed preprocessing pipeline, which consists of resizing, normalization, batch-based loading was performed to prepare the DEX images for deep feature extraction, handcrafted feature computation and machine learning experiments described in the following sections.

3.2 Image features for malware detection

In addition to Transfer learning-based feature embeddings, a variety of handcrafted image descriptors are used to extract complementary structural and texture-based information from the DEX-derived grayscale images. This hand-crafted information is considered to encode local intensity variations, gradient orientations, spatial structures and keypoint-based features which may not be sufficiently captured in deep semantic embeddings. The combination of handcrafted and deep features improves the robustness of the overall representation and allows them to better distinguish benign apps from malicious ones. Various handcrafted features used in this work as described below. Histogram of Oriented Gradients (HOG) for capturing the distribution and orientation of gradients in regions that arise out of repetitious bytecode patterns or code obfuscation.

The salient interest points preserving the invariance of in scale, rotation and affine transformation are detected by SIFT and ORB descriptors, thus structural irregularities or similar signatures shared across a specific type of threat can be discovered. Local Binary Patterns (LBP) extract micro-texture representations by encoding the relationship between a pixel and its neighboring intensities, which can be effective to capture slight textures variation due to bytecode compilation behavior. Lastly, texture features derived from the Gray-Level Co-occurrence Matrix (GLCM) measure second-order spatial statistics such as contrast, homogeneity, correlation, and energy, reflecting the global texture structure of DEX images.

These handcrafted descriptions enrich complementary local patterns and structure cues to significantly improve the fused space with deep embeddings provided by CNN or transfer learning models. Such complementary combination of feature types enables a more holistic encoding of Android executable code that is utilized by the hybrid feature concatenation.

4. Results and discussion

This section presents the experimental workflow, model evaluations and quantitative results obtained using the DEX-image dataset. The proposed work is divided into different segments such as CNN based classification, Transfer Learning models classification, feature extraction methods, hybrid feature fusion and machine learning classification over fused features. For all transfer learning experiments, the pretrained backbone networks are initialized with ImageNet weights and keep fixed as feature extractors. The convolutional layers of DenseNet121, MobileNetV2 and InceptionV3 were maintained as non-trainable to retain the already obtained generic visual representations and thereby also to avoid overfitting in relation to our moderate dataset size.

4.1 Implementing CNNs

To provide the benchmark for DEX-image–based feature classification of malware, a CNN is trained on grayscale images. The model was composed of 3 × 3 size convolutional layers and ReLU activation, then max-pooling and fully connected dense layers for binary classification. A sigmoid activation was applied at the output. The network was trained for 30 epochs via Adam optimization with a learning rate of 0.001, binary cross-entropy loss and batch size of 32. The CNN model obtained test accuracy of 90%. This result indicates that the spatial and structural patterns in DEX images are discriminative enough to be used for malware identification even without additional feature extraction or transfer learning. Figure 2 and Figure 3 shows epoch wise accuracy and loss with CNN model.

4.2 Implementing DenseNet

DenseNet121 was tested on the DEX-image dataset to determine its ability for learning discriminative visual patterns of executable bytecode. Architecture is able to benefit from the dense convolutional connections which allows for efficient re-usage of features and gradient backpropagation and helps in capturing both the fine-grained as well as high-level structural features existing in DEX-derived images.

The training and validation accuracy curves across epochs are shown in Figure 4. The validation accuracy rises rapidly in the first few epochs and then remains stable, which demonstrates that DenseNet121 can learn informative structural patterns from the DEX images. The trend of training accuracy is also keeping a smooth upward moving, meaning the model generalizes well and doesn't have overfitting. The epoch-wise loss for training and validation datasets is shown in Figure 5. The test accuracy of DenseNet121 is 0.8869, which is slightly lower than the baseline performance of CNN (90%). But DenseNet can still behave well and acquire more robust features from the Dex images. The stable convergence patterns observed in the accuracy curve validate DenseNet's capability to model executable-level structural properties effectively.

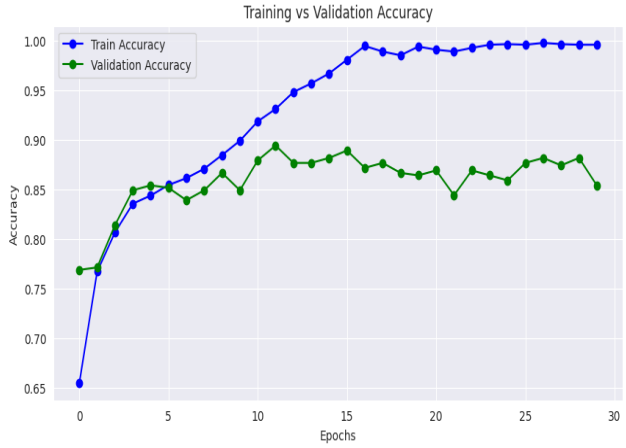

Figure 2. Epoch wise accuracy with CNN

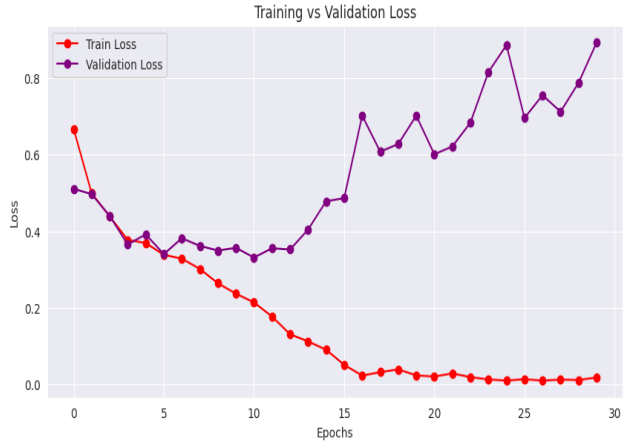

Figure 3. Epoch wise loss with CNN

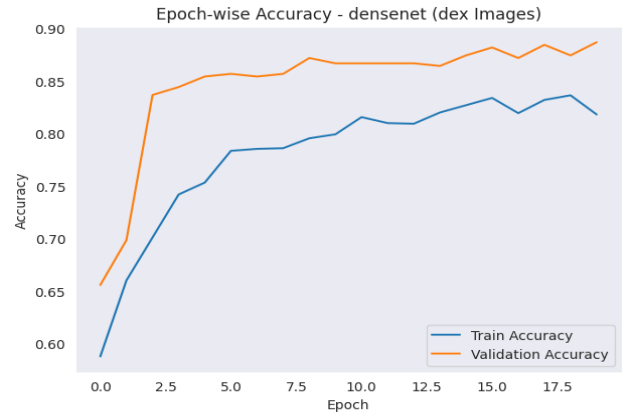

Figure 4. Epoch wise accuracy with DenseNet

4.3 Implementing MobileNet

MobileNetV2 was also tested on the DEX-image dataset to study a lightweight deep architecture that is engineered for efficient feature extraction. The model is based on an inverted residual structure where the input and output of the residual block are thin bottleneck layers 1 such that we can use low-dimensional representations in the middle. Hence, MobileNet is appropriate for the large-scale or resource limited malware detection problems. Figure 6 shows the learning curve of the accuracy for each epoch on training and validation sets. Figure 7 shows the loss per epoch over both the training and validation sets. MobileNetV2 obtained a test accuracy of 0.8919. The stable and competitive performance also justifies that it is appropriate to be used as a lightweight feature extract in the entire malware detection system.

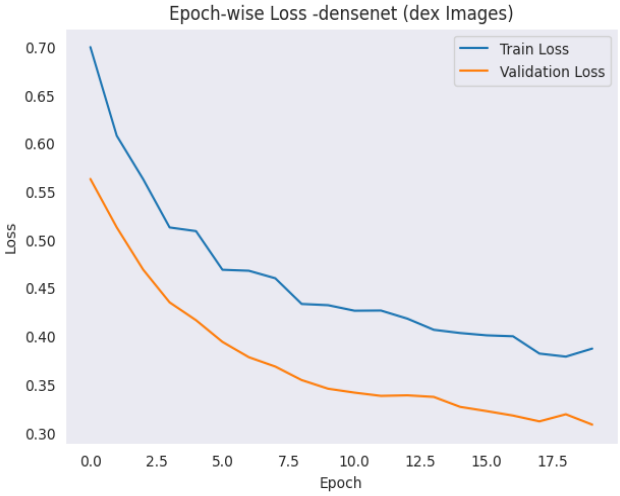

Figure 5. Epoch wise loss with DenseNet

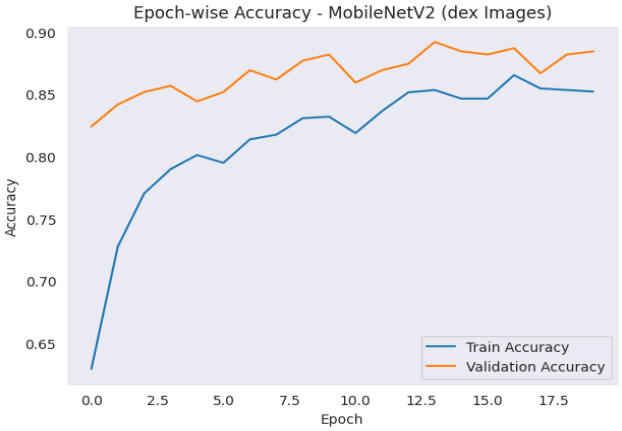

Figure 6. Epoch wise accuracy with MobileNet

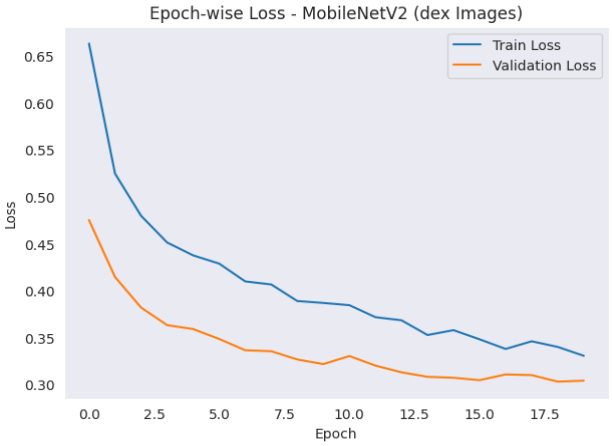

Figure 7. Epoch wise loss with MobileNet

4.4 Implementing inception

Inception-V3 is also tested on DEX-image dataset to compare with existing methods. InceptionV3 extracts input features using parallel convolutional paths with various kernel sizes, which is helpful to represent multi-scale spatial information of the bytecode-based images. This makes the method very well-suited for detecting both nuances of local texture and global structural patterns found in malware and non-malware. Epoch-wise accuracy and loss curves for InceptionV3 are shown in Figure 8 and Figure 9, respectively. The training and validation curves of accuracy show good learning behavior, as they are consistently increasing from one epoch to another. The curves stay close to each other, also suggesting that only minor overfitting is present and the model generalizes well for unobserved DEX samples. The validation accuracy becomes plateau very early, indicating that InceptionV3 can mine meaningful and stable feature representations through DEX images. The final accuracy achieved on the test set is 0.864.

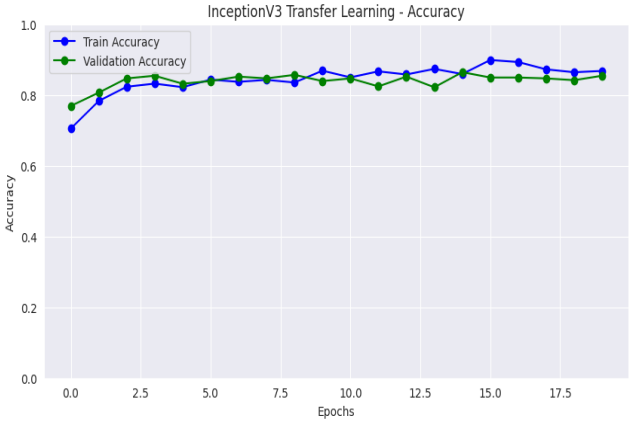

Figure 8. Epoch wise accuracy with inception

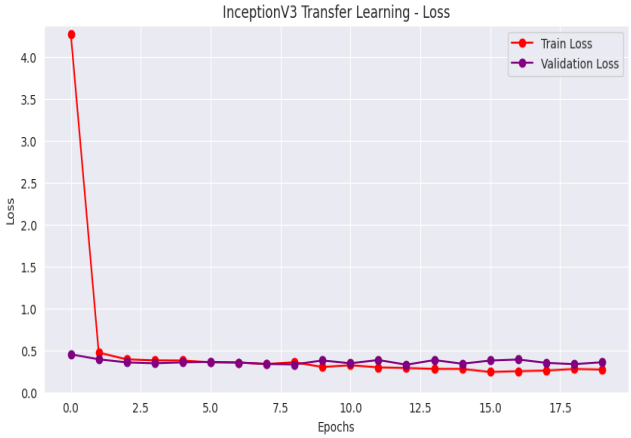

Figure 9. Epoch wise loss with inception

4.5 Transfer learning feature extraction

For high-level semantics representation of the DEX-image dataset, multi-model Transfer Learning (TL) was adopted to extract deep feature vectors. Rather than training all these models end-to-end for classification, we utilized the pretrained networks as feature extractors by discarding their fully connected classification layers and keeping only the convolutional body. All DEX images were fed into the respective TL model and global average pooling layer after which the extracted deep feature representation was written as output. This process aimed at producing discriminatory feature embeddings that would capture structural, spatial and hierarchical arrangements existing in DEX bytecode images. DenseNet121, MobileNetV2 and InceptionV3 generated feature vectors of different lengths to each other in consideration of the architectural variations between the networks. The DenseNet121 feature descriptors were rich in connectivity-based abstractions, MobileNetV2 generated low-weight compact embeddings that could be efficiently classified and InceptionV3 provided multi-scale semantic information from parallel convolutional paths. The TL feature vectors were the base for generating the hybrid features set exploited in other machine learning experiments. Such representations were high-level semantical capsules that are consistent with handcrafted descriptors, and they greatly improve the discriminative power of the final fused feature space.

4.6 Handcrafted feature extraction

In addition to deep semantic representations obtained through Transfer Learning models, several hand-crafted descriptors were extracted from the DEX-image dataset at both patch and image levels to model complementary low-level visual properties alongside the deep semantic representations acquired with Transfer Learning models. Handcrafted ones are effective in capturing fine-grained textural, gradient and key point-based cues which may not be well represented in deep feature embeddings. Their incorporation increases the discriminative ability of the hybrid feature space for malware classification. HOG was used to capture direction gradient patterns and edge distributions – common representations of structural modifications performed by malicious code. Keypoint descriptors, such as SIFT (Scale-Invaraint Feature Transform) and ORB (ORiented FAST and Rotated BRIEF), were used to find stable local interest points inside the DEX images and describe them. Such descriptors can be used to emphasize local anomalies, common code regions, and transformation-invariant patterns that are often presented in malicious samples.

Descriptors oriented to texture were also computed for statistical relationships among the intensities of pixels. Local Binary Patterns (LBP) computed to represent microtextured variations by encoding local neighborhood intensity differences and Gray-Level Co-occurrence Matrix (GLCM) characteristics which measured of the spatial co-occurrence properties i.e., contrast, homogeneity, energy and correlation. These texture descriptors are appropriate to detect subtle structure patterns appeared in various malware families. The fusion of the gradient-based, key point-based as well as texture-based handcrafted features provides a rich set of complementary information. When combined with Transfer Learning embeddings, these tailored descriptors add discriminative clues for making a decision of whether an application is benign or malicious.

Fixed parameter settings were used for all handcrafted feature extractors. HOG descriptors used with 9 orientation bins, cell size in terms of pixel values (8×8) and 2×2 cells per block. LBP were calculated using a radius of 1 and eight neighboring pixels. GLCM texture features were computed with a pixel offset distance of one and four directions ($\theta = 0°$, 45°, 90° and 135°) to represent the spatial relationship among intensity. SIFT key points were computed with OpenCV default settings, and ORB features with five hundred key points per image. These fixed parameters have allowed for the same feature extraction across all DEX images.

4.7 Hybrid Feature Fusion

A feature-level fusion strategy was implemented to increase the discriminative capability of the malware detection system by integrating deep semantic features obtained from Transfer Learning models with handcrafted descriptors. For each DEX image, high-level embeddings generated from DenseNet121, MobileNetV2 and InceptionV3 were obtained as well as low-level gradient, key point and texture descriptors including HOG, SIFT, ORB, LBP GLCM. These vectors were concatenated together to form a single feature representation comprising both global and local information from the DEX images. Intuitively, this fusion is necessary because deep features and handcrafted features contain complementary information. Transfer Learning architectures can capture hierarchical semantic abstractions and, hence, effectively learn generic spatial relationships and patterns present in malware bytecode. But they could be unsuccessful on localized features or micro-textures. On the other hand, manually designed descriptors focus on pixel-level fluctuations, directional gradients, local textures and discriminative key points which can model obfuscated areas, inserted code or copied malicious patterns. The combination of these two categories provides a more complete and comprehensive view over the underlying executable code.

The deep model output 1026 features for DenseNet121, 1282 for MobileNetV2 and 2050 for InceptionV3. Manual extraction methods extract26246 HOG features, 6402 SIFT features, 1602 ORB features, 61 LBP features and 8 GLCM feature. All these properties are appended into a single vector and machine learning classification in based on that. Prior to classification, the fused feature matrix was aligned and normalized to ensure consistency across feature sources. The holistic embedding was able to serve as strong input for the subsequent ML models in the following subsection. The experimental results demonstrate that: fusing both deep and handcrafted descriptors greatly enhance the classification performance than employing a single feature type, which indicates the effectiveness of hybrid feature learning in DEX-based Android malware detection.

4.8 Machine learning classification using fused DEX features

The concatenated feature maps of Transfer Learning embeddings and handcrafted descriptors were applied to several ML classifiers in order to evaluate their performance for Android malware detection. All of the models were trained on the hybrid feature matrix from the DEX-image dataset and accuracy was calculated for test set. Results with ML models are presented in Table 1 and Figure 10.

Table 1. Results with ML models and fused features

Model	Accuracy
Random Forest	94.4%
Logistic Regression	94.3%
SVM	93.7%
KNN	89.3%
Naive Bayes	73.9%

Random Forest performed the best among all the classifiers in terms of accuracy, that summarized to 94.40%. The Random Forest classifier was trained with 100 decision trees and node splitting criterion was the Gini impurity. Logistic Regression did extremely well with accuracy of 94.33%. Logistic Regression model was fitted with L2 penalty and C value of 1.0. For stable optimization, we use the liblinear solver for binary classification in high-order combined feature vectors. The accuracy of SVM with RBF kernel is 0.9371 again, indicating that the fused information can support margin-based classification. The good performance of SVM demonstrates that the two types of descriptors have a uniform distribution with high discrimination. In addition, KNN achieved an accuracy of 0.8936, which is a moderate performance. With distance as its metric, KNN is more susceptible to feature scaling and redundancy that could have contributed to kNN's relatively lower accuracy compared with the top models. Naïve Bayes achieved a lower performance with an accuracy of 0.7398.

Its assumption of independence between features as well as the normality constraint may hinder it for modeling more complicated, correlated patterns in concatenated feature vectors. On the whole, hybrid feature representation provided a better and steady classification performance improvement on all tested machine learning models, in which Random Forest and Logistic Regression were identified as the best two models for DEX-based Android malware detection. While the performance gaps between CNN, Transfer Learning models and machine learning classifiers are not large, there were consistent improvements with each evaluation scenario when using hybrid feature fusion. No formal testing for statistical significance, such as paired t-tests or bootstrapping, was performed due to the use of a single stratified train–test split. However, similar performance gains on a repeated basis across model families suggest that the trends are not random but instead stable and systematic. Since the dataset is class-balanced and tested with several independent learning models, stable performance across CNN, transfer learning and machine learning classifiers suggest that the models are not clearly overfitting. However, additional validation on a larger and more diverse test set would help to increase the applicability of proposed approach.

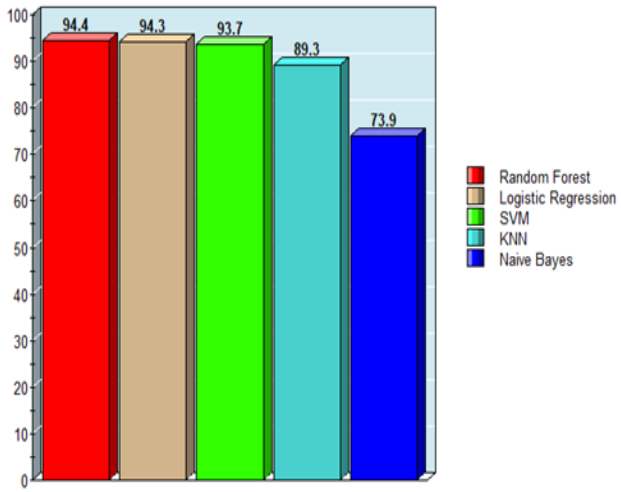

Figure 10. Results with ML models and fused features

Table 2 shows comparison of proposed model with existing works. traditional feature selection machine learning and deep learning-based CNN-RNN models are able to achieve moderate performance. On the other hand, the proposed hybrid DEX-image-based framework achieves an accuracy of 94.4%, showing the benefits of using both deep and handcrafted features. This comparison demonstrates the superior performance of the proposed method to some recent related works. However, there are some limitations of the proposed method despite its good performance. As the approach is based on image-like, static DEX bytecode analysis, it could be sensitive to sophisticated adversarial pouring and packing methods that can mutate bytecode structure considerably. Moreover, the dataset is collected from public repositories, probably bringing data bias and restricted generalization to real-world scenarios. Future work aims to address these inabilities with larger datasets, adversarial robustness investigation, and also hybrid static–dynamic methods.

Table 2. Comparison with existing works

Model	Accuracy
XGBoost [22]	88%
Feature selection +ML Model [23]	92.1%
CNN+RNN [24]	89.14%
Proposed Method	94.4%

5. Conclusion

This work presented a DEX-image–driven hybrid framework for Android malware detection that integrates deep feature embeddings with handcrafted image descriptors and machine learning classification. By converting the classes. Dex file of each application into a grayscale image, the proposed approach captures structural, spatial and texture-level patterns that are difficult to conceal through traditional code obfuscation techniques. Deep features extracted using CNN and three state-of-the-art Transfer Learning models—DenseNet121, MobileNetV2 and InceptionV3—were complemented by handcrafted descriptors such as HOG, SIFT, ORB, LBP and GLCM to form a comprehensive representation of executable-level behavior. Experimental results demonstrated that the deep models provide strong standalone performance, with the baseline CNN achieving 90% accuracy and the Transfer Learning architectures producing competitive results on the DEX-image dataset. When combined with handcrafted features, the fused representation delivered substantial performance gains across all machine learning classifiers. The Random Forest and Logistic Regression models achieved the highest accuracies of 94.40% and 94.33%, respectively, confirming the effectiveness of hybrid feature learning for malware detection. These findings highlight the advantage of using DEX-based visual representations and multi-source feature fusion over traditional single-feature approaches. Overall, the proposed framework offers a robust, scalable and obfuscation-resilient solution for Android malware identification. The future work will aim to extend the exploration of more Transfer Learning models architectures with more advanced Vision Transformers, multi-modal network fusion using opcodes, API-calls and other features.

Ethical issue

The authors are aware of and comply with best practices in publication ethics, specifically regarding authorship (avoidance of guest authorship), dual submission, manipulation of figures, competing interests, and compliance with research ethics policies. The authors adhere to publication requirements that the submitted work is original and has not been published elsewhere.

Data availability statement

The manuscript contains all the data. However, more data will be available upon request from the authors.

Conflict of interest

The authors declare no potential conflict of interest.

References

[1] R. Hasan et al., “Enhancing malware detection with feature selection and scaling techniques using machine learning models,” Sci Rep, vol. 15, no. 1, Mar. 2025, doi: 10.1038/s41598-025-93447-x.

[2] A. Pathak, Th. S. Kumar, and U. Barman, “Static analysis framework for permission-based dataset generation and android malware detection using machine learning,” EURASIP J. on Info. Security, vol. 2024, no. 1, Oct. 2024, doi: 10.1186/s13635-024-00182-3.

[3] A. Mahindru et al., “PermDroid a framework developed using proposed feature selection approach and machine learning techniques for Android malware detection,” Sci Rep, vol. 14, no. 1, May 2024, doi: 10.1038/s41598-024-60982-y.

[4] A. Kaur, S. Lal, S. Goel, M. Pandey, and A. Agarwal, “Android Malware Detection System using Machine Learning,” Proceedings of the 2024 Sixteenth International Conference on Contemporary Computing. ACM, pp. 186–191, Aug. 08, 2024. doi: 10.1145/3675888.3676049. https://doi.org/10.1109/TITS.2014.2345663.

[5] M. U. Rashid et al., “Hybrid Android Malware Detection and Classification Using Deep Neural Networks,” Int J Comput Intell Syst, vol. 18, no. 1, Mar. 2025, doi: 10.1007/s44196-025-00783-x.

[6] M. Aamir et al., “AMDDLmodel: Android smartphones malware detection using deep learning model,” PLoS ONE, vol. 19, no. 1, p. e0296722, Jan. 2024, doi: 10.1371/journal.pone.0296722.

[7] A.Sonya and R.Ram Deepak, “Android Malware Detection and Classification Using Machine Learning Algorithm”, Int. j. commun. netw. inf. secur., vol. 16, no. 4, pp. 327–347, Sep. 2024, https://ijcnis.org/index.php/ijcnis/article/view/7012.

[8] A. Pathak, U. Barman, and Th. S. Kumar, “Machine learning approach to detect android malware using feature-selection based on feature importance score,” Journal of Engineering Research, vol. 13, no. 2, pp. 712–720, June 2025, doi: 10.1016/j.jer.2024.04.008.

[9] K. A. Ahmed, K. Boopalan, K. Lokeshwaran, R. Sugumar, and C. Kotteeswaran, “Analysis of android malware detection using machine learning techniques,” AIP Conference Proceedings, vol. 3063.

AIP Publishing, p. 020005, 2024. doi: 10.1063/5.0199036.

[10] M. A. Haq and M. Khuthaylah, "Leveraging Machine Learning for Android Malware Analysis: Insights from Static and Dynamic Techniques," Eng. Technol. Appl. Sci. Res., vol. 14, no. 4, pp. 15027–15032, Aug. 2024, doi: 10.48084/etasr.7632.

[11] A. O. Christiana, B. A. Gyunka, and A. Noah, "Android Malware Detection through Machine Learning Techniques: A Review," Int. J. Onl. Eng., vol. 16, no. 02, pp. 14–30, Feb. 2020, doi: 10.3991/ijoe.v16i02.11549.

[12] J. Lee, H. Jang, S. Ha, and Y. Yoon, "Android Malware Detection Using Machine Learning with Feature Selection Based on the Genetic Algorithm," Mathematics, vol. 9, no. 21, p. 2813, Nov. 2021, doi: 10.3390/math9212813.

[13] C. Palma, A. Ferreira, and M. Figueiredo, "Explainable Machine Learning for Malware Detection on Android Applications," Information, vol. 15, no. 1, p. 25, Jan. 2024, doi: 10.3390/info15010025.

[14] M. N.-U.-R. Chowdhury, A. Haque, H. Soliman, M. S. Hossen, T. Fatima, and I. Ahmed, "Android Malware Detection using Machine learning: A Review," 2023, arXiv. doi: 10.48550/ARXIV.2307.02412.

[15] E. Odat and Q. M. Yaseen, "A Novel Machine Learning Approach for Android Malware Detection Based on the Co-Existence of Features," in IEEE Access, vol. 11, pp. 15471-15484, 2023, doi: 10.1109/ACCESS.2023.3244656.

[16] D. A. Kumar et al., "Machine Learning Approach for Malware Detection and Classification Using Malware Analysis Framework", Int J Intell Syst Appl Eng, vol. 11, no. 1, pp. 330–338, Feb. 2023, https://ijisae.org/index.php/IJISAE/article/view/2543.

[17] A. Lakshmanarao and M. Shashi, "Android Malware Detection with Deep Learning using RNN from Opcode Sequences," International Journal of Interactive Mobile Technologies., vol. 16, no. 01, pp. 145–157, Jan. 2022, doi: 10.3991/ijim.v16i01.26433.

[18] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android Malware Detection Based on a Hybrid Deep Learning Model," Security and Communication Networks, vol. 2020, pp. 1–11, Aug. 2020, doi: 10.1155/2020/8863617.

[19] Navaneethan S et al.., "ScanSavant: Malware Detection for Android Applications with Explainable AI," Int. J. Interact. Mob. Technol., vol. 18, no. 19, pp. 171–181, Oct. 2024, doi: 10.3991/ijim.v18i19.49437.

[20] H. AlOmari, Q. M. Yaseen, and M. A. Al-Betar, "A Comparative Analysis of Machine Learning Algorithms for Android Malware Detection," Procedia Computer Science, vol. 220, pp. 763–768, 2023, doi: 10.1016/j.procs.2023.03.101.

[21] M. Kavitha and M. U. Rani, "A Deep Learning-Driven Image Transformation Approach for Android Malware Detection using CNN and Transfer Learning Techniques," 2025 5th International Conference on Pervasive Computing and Social Networking (ICPCSN), Salem, India, 2025, pp. 1503-1508, doi: 10.1109/ICPCSN65854.2025.11034903.

[22] S. Aurangzeb, M. Aleem, M. T. Khan, G. Loukas, and G. Sakellari, "AndroDex: Android Dex Images of Obfuscated Malware," Sci Data, vol. 11, no. 1, Feb. 2024, doi: 10.1038/s41597-024-03027-3.

[23] K. S Ranadheer Kumar et al., "Enhancing Android Malware Detection through Filter-Based Feature Selection and Machine Learning Classification," Journal of Electrical Systems,Volume.20 No.3, 2024, https://journal.esrgroups.org/jes/article/view/4528

[24] S. Kotha, H. S, S. D, and S. Ahmed, "Android Malware Detection Using Deep Learning," 2025 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC). IEEE, pp. 1–4, May 16, 2025. doi: 10.1109/assic64892.2025.11157977.