Journal of Information Systems Engineering and Management
2025, 10(15s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Feature Extraction and Classification in Android Malware Detection with the Integration of Autoencoders and Transfer Learning

Arun .N [1], T. R. Nisha Dayana[2]
[1,2] Department of Computer Science, Vel's Institute of Science, Technology & Advanced Studies, Chennai, Tamil Nadu, India

ARTICLE INFO

Received: 07 Dec 2024
Revised: 22 Jan 2025
Accepted: 07 Feb 2025

ABSTRACT

Introduction: Mobile security suffers greatly by the quick spread of Android malware, leading to the need for sophisticated detection methods that can change to meet new threats. Malware is still a difficult security issue in the Android ecosystem because it frequently obfuscates itself to avoid detection. Semantic behavior feature extraction is essential in this situation in order to build a reliable malware detection model

Objectives: To provide an overview of android malware, including the impact of malware detection, the significance of malware detection, its types, and a framework for detecting Android malware that uses Transfer Learning (TL) to forecast malware in the Android ecosystem and AEs (AutoEncoders) to extract features.

Methods: This study presents an integrated deep learning (DL) method for Android malware detection that combines AEs for feature extraction and TL for classification. In order to effectively depict both benign and malevolent actions, AEs are used to extract latent, highly dimensional features from both static and dynamic analytical data. Then, TL uses deep neural networks that have already been trained to identify Android apps more accurately and with less training time

Results: The tests were conducted on three datasets with two labels in the "class" attribute "0" for benign and "1" for malicious in order to evaluate the effectiveness of the suggested framework. With an enhanced MAE value of 0.001 and RMSE value of 0.063 attain an 99.99% accuracy. The findings show that the proposed model achieved remarkable accuracy and has the potential to produce reliable malware detection results

Conclusions: An integrated DL strategy for Android malware detection that combines AEs for feature extraction and TL for classification has been presented. AEs are used to extract high-dimensional, latent characteristics from both static (code-related) and dynamic (behavior-related) Android app analysis data. The system may thus effectively capture and reflect both beneficial and harmful behaviors.

Keywords: Android Malware Detection, Feature Extraction, Transfer Learning, AutoEncoders

INTRODUCTION

The Android ecosystem has grown rapidly, making it an influential platform in the mobile industry. But this appeal has attracted bad actors, which has resulted in an unprecedented rise in Android malware. Android operating systems (OS) are widely used because of their user-friendly design and open-source nature. In order to reduce the danger of various malware attacks, a malware detection system for Android OS devices must be designed. This renders them susceptible to a variety of security threats (Joshi et al,.2023).Threats from these harmful apps include financial fraud, data theft, privacy violations, and illegal device control.

The dynamic character of Android malware poses a serious threat to conventional malware detection methods as it changes over time using strategies like concealment, polymorphism, and zero-day weaknesses. Malware detection systems have faced significant strain and challenges in recent years due to a significant increase in both the quantity and variety of Android malware (Liu et al., 2024). Signature-based and heuristic approaches, which are currently used for Android malware detection, frequently find it difficult to keep pace with the quick appearance of new and complex ransomware variants. Furthermore, there is an urgent need for sophisticated feature extraction and classification techniques that can successfully spot risky patterns due to the increasing number and complexity of Android applications (Roy et al., 2023).

Some of the types of android malware include the following

- Trojan horses are malicious programs that pose as trustworthy apps in order to fool users into installing them. It gives the attacker illegal access to private data, including financial information and login passwords.
- Ransomware encrypts or locks a user's data or device and then demands money to unlock it. A ransom notification appears on the device's screen. It makes data unavailable by encrypting them.
- Spyware secretly tracks user behavior and transmits information to the hacker. In addition to tracking SMS, calls, GPS position, and app activity, it can record users by turning on a microphone or camera.
- Adware shows invasive commercials to make money for the attackers. It floods the device with banner ads, frequently rerouting users to malicious websites and potentially degrading device performance.
- Worms are malware that replicates itself and spreads without human intervention.It spreads among devices by taking advantage of flaws in the Android operating system.
- It degrades performance by using up device resources.
- Keyloggers record the keystrokes to get private data, such as PINs and passwords. It transmits the attacker the keystrokes that have been collected while running silently in the background.

DL methods have demonstrated enormous promise in tackling these issues, especially AEs and TL. While TL uses pre-trained models to enhance classification performance, even with small labeled datasets, AEs allow the extraction of concealed, high-dimensional traits that capture complex correlations and patterns in malware activity. The necessity to create a strong, scalable, and adaptable malware detection system that is capable of handling the complexity of contemporary Android spyware is what spurred this study. This study intends to improve detection accuracy, lower computational cost, and offer a framework that can adjust to changing malware threats by combining the advantages of AEs with TL. The ultimate objective is to close significant holes in existing detection techniques in order to contribute towards a safer Android environment.

Android Malware Detection

The term "Android malware" describes malicious apps that can do a number of detrimental things to Android devices and users, including encrypting or destroying data, leaking private user information, stealing user credentials, and altering device settings. As the OS of choice in the smartphone market, Android has grown to be a top target for cybercriminals. Malicious actors now have more ways to produce and disseminate a variety of Android malware due to the development of sophisticated tactics(Manzil t al., 2024). Machine learning-based techniques have replaced signature-based techniques as the primary means of detecting Android malware. Feature extraction is crucial to machine learning techniques. The methods to detect Android malware may be divided into groups according to the various feature extraction analysis techniques or methods. The methods for malware analysis includes the following (kumar et al., 2024)

Static Analysis: It examines the APK (Android Package) or application's code without running it. Methods include examining code structure, permissions, and API calls. Although static analysis is quick, it may be avoided by using encryption and obfuscation.

Dynamic Analysis: Examines how an application behaves when running in a regulated setting, such as a sandbox. It records runtime operations such as memory utilization, file system interactions, and network connectivity. Although it requires a lot of resources, dynamic analysis can identify behavior-based malware. The detection techniques includes

Signature-Based Detection: Identifies patterns or signatures of known malware within the application. It works well against known threats, but it is ineffective against malware that has been altered or created recently (such as zero-day threats).

Behavior-Based Detection: The goal of behavior-based detection is to find suspicious patterns or unusual activities that point to malevolent intent.

Machine Learning-Based Detection: This method divides apps into harmful and benign categories using supervised, unsupervised, or deep learning algorithms.

Deep Learning Techniques : Using sophisticated neural networks such as CNNs, RNNs, or transformers, deep learning techniques automatically extract features and increase detection accuracy(Bean et al., 2024).

Feature Extraction

The basis of every Android malware detection solution is feature extraction. These systems' scalability, accuracy, and resilience might be improved by utilizing cutting-edge strategies like DL and hybrid approaches. Sophisticated techniques are required to conduct thorough investigations and extract aspects of Android malware. A dataset's core is represented by its characteristics. Every column in a tabularly organized dataset constitutes a feature, which may be used as an input in any model. Variables or properties that are statically or dynamically extracted from apps are regarded as features in the wider context of the Android malware ecosystem. These features fall into two groups: static and dynamic. Techniques based on static analysis are used to extract features from the files that make up app installation packages. Strategies based on dynamic analysis mostly concentrate on network or system traffic data while applications are operating. These methods need running the applications, which adds overhead and discomfort. Therefore, the majority of methods use static analysis to extract features (Liu et al., 2024).

The most thorough features are hybrid ones, which examine applications from several angles. Both static and dynamic characteristics that are crucial for malware identification

Static features: During static analysis, Android files are examined for requests for permissions, opcode sequences, API calls, and other information. Static detection is popular in the realm of Android malware detection since it offers a lot of easily extracted optional characteristics.

Dynamic features: Dynamic analysis entails running the program in a sandbox environment and tracking the behavior of the program's system call, network traffic, CPU data, and API call sequence in order to observe data movement during the program's running phase and track the behavior of the program processing closer to the real behavior.

Transfer Learning in Malware Detection

Transfer learning is used to identify Android malware because it transfers known properties from a trained source model to a target model, minimizing the requirement for large quantities of processing resources and additional training data. The final few dataset-specific layers must be learned, which drastically reduces the time necessary to create and train a new model.It takes a lot of processing resources to train DL models using complicated and mixed datasets. This large computing cost can be decreased by using the TL technique.Overfitting issues occur when traditional DL and ML techniques are created using an insufficient dataset. TL is used to fix the issue by optimizing the model layers(Raza et al., 2024).

According to the research, the majority of feature selection techniques and detection models currently in use for malware analysis and detection have a high false alarm rate and poor detection accuracy. Therefore, it remains an active research topic and a significant challenge to build an appropriate feature selection technique that may increase the rate of detection, precision in classification of new malware variants, and reduction of irrelevant features from large amounts of information.

Firoz et al.,(2024) introduce a novel method for malware detection that makes use of AEs. The approach employs a hybrid DL model that combines convolutional neural networks (CNN) for malware classification with AEs for feature extraction. Researchers extracted properties from noisy input using autoencoders. This effectively reduces the number of dimensions in the huge feature collection and produces groups of incorrect predictions through reconstruction error. ML and DL models then use the newly rebuilt input information to make predictions.

Joshi et al., suggested a ML based malware detection model that uses stacking to identify malware on Android devices. The model creation procedure makes use of four distinct ML models: Random Forest(RF), Histogram Gradient Boosting, Catboost, and Support Vector Machine(SVM). The two latest datasets, CIC-MalDroid 2020 and CIC-MalMem 2022, are used .

Liu et al., (2024) introduces SeGDroid, a revolutionary approach to Android malware detection that relies on extracting semantic information from vulnerable function call graphs (FCGs). In particular, we develop a graph pruning technique to construct a sensitive FCG from an initial FCG. The technique eliminates the unnecessary FCG nodes while maintaining the sensitive API (security-related API) call context.

Aldhafferi et al., (2024) developed an extensive strategy that included feature preprocessing and normalization, dynamic feature extraction from Android apps, and the use of SVR with a Radial Basis Function (RBF) kernel for malware classification. The efficiency of the suggested model in differentiating between malicious and benign apps was greatly increased by its ability to capture intricate, non-linear interactions in the feature space.

Ullah et al., (2024) develops the network-based Android malware detection and classification method known as NMal-Droid. In order to filter the combination of HTTP traces and TCP flows from PCAPs (Packet Capturing) files, we first created a packet parser algorithm. Second, the fine-tune embedding method is created, which analyzes feature embeddings in three distinct ways—random, static, and dynamic—using a word2vec pre-trained model. Feature-matrix matrices with linked meanings may be learned and extracted using it. Third, useful characteristics are extracted from embedded data using a CNN. Fourth, gradient computation is intended to be performed in both time-forward and time-reversed scenarios using the Bi-directional Gated Recurrent Unit (Bi-GRU) neural network. Lastly, a CNN-BiGRU multi-head ensemble is created for precise malware identification and categorization.

Shen et al. (2023) provide GAResNet (ResNet with Group Convolution and Attention), a malware detection system for Android that uses TL. Furthermore, incorporated a group convolution and attention strategies into the original residual network model in order to extract more effective and distinctive characteristics that will further improve the malware detection capabilities for new malware. More specifically, the model migrates to the Android dataset after training on the Microsoft dataset.

Tasyurek et al.,(2024) presented a CNN model called Real Time-Droid (RT-Droid) based on YOLO V5 and has a very quick malware detection capacity. With a high degree of accuracy, RT-Droid can identify Android malware almost instantly. The initial step in this procedure is to extract the features from the Android manifest file and turn them into an RGB picture that resembles a QR code. CNN-based DL models are therefore used to process images. Using the TL approach, those images were utilized to train the VGGNet, Faster R-CNN, and YOLO V4 and V5 models.

Bostani presents EvadeDroid, an adversarial problem-space approach that is intended to successfully elude black-box Android malware detection in practical situations. EvadeDroid uses an n-gram-based method to build a set of problem-space modifications from benign donors that are comparable to malware applications at the opcode level. Using a gradual and repetitive modification technique, these changes are subsequently employed to turn malware occurrences into benign ones.

Ghourabi, et al., (2024) suggests a novel method for detecting and classifying Android malware that pinpoints the key characteristics to enhance classification accuracy and cut down on training time. There are two primary phases in the suggested method. The most crucial characteristics are first chosen using a feature selection technique based on the Attention mechanism. The Android samples are then classified and the malware is identified using an improved Light Gradient Boosting Machine (LightGBM) classifier.

Atacak et al., (2022) suggests an architecture using fuzzy logic's ability to make decisions with the CNNs capacity for feature extraction. The approach makes the feature size appropriate for ANFIS input while extracting features from permission information using a limited number of filters and convolutional layers. Furthermore, it permits the permission data to influence the categorization without being overlooked.

The most comprehensive surveys are restricted to static, dynamic, and hybrid methods or malware detections based on machine learning. Furthermore, the majority of them have not sufficiently addressed the typical difficulties in this subject. Because of this, there is still a research void in this area. This encourages us to thoroughly examine current detection methods in order to draw objective conclusions and debate various viewpoints in order to offer insightful recommendations. An analysis of this kind could be useful in pointing out areas of inadequacy and suggesting possible directions for further study. Researchers may be able to learn more about the topic as a result of such findings and recommendations.

OBJECTIVES

The main Objective of the study is to

- Present a detailed background of malware, effect of malware detection, importance of malware detection, type of malware, various malware evasion techniques and taxonomy of malware analysis.
- Review the methods applied in the development of malware detection systems
- Design a framework for android malware detection using AEs for feature extraction and TL for prediction of malwares in the android ecosystem.

METHODS

Figure 1 shows a high-level process for detecting Android malware that makes use of AEs and TL. An Android dataset with samples of both malicious and benign apps serves as the starting point for the entire process. Features like permissions, API requests, or system activity logs are frequently included in these datasets. Dimensionality reduction and the extraction of significant features from unprocessed data are accomplished by use of an AE. AEs encode input data into the latent space and then reconstruct it to learn concise representations of the data. The retrieved features are used to fine-tune a MobileNetv3 model that has already been trained on an identical task. In order to identify if the sample is malicious or not, this step classifies it. Standard measures including accuracy, precision, recall, and F1-score are used to assess the predictions. The end result shows if the Android app is categorized as benign or malicious.

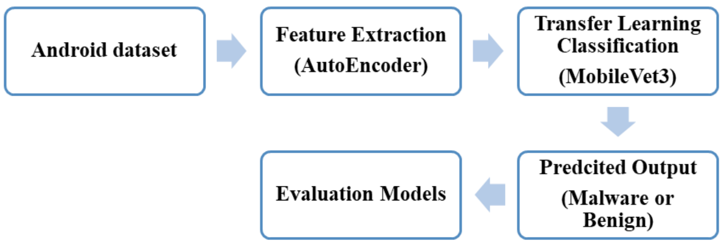
Android dataset
Feature Extraction (AutoEncoder)
Transfer Learning Classification (MobileVet3)
Predcited Output (Malware or Benign)
Evaluation Models

Figure 1 Workflow for malware detection

This process demonstrates how to effectively and precisely identify Android malware by using AEs for feature extraction and MobileNetv3 for TL.

Auto Encoders:

AEs are unsupervised neural networks that are capable of learning latent features, or compacted, meaningful depictions, from input data. AEs are helpful for feature extraction in Android malware detection, particularly when dealing with high-dimensional data like as permissions, API calls, opcode sequences, or behavioral logs. AEs remove noisy or unnecessary features from raw data and shrink highly dimensional feature vectors while maintaining important information.

Additionally, it finds hidden trends in the data, which aids in differentiating between harmful and benign actions. The architecture of AE is presented in figure 2. The input layer corresponds to the dimensionality of the data being entered, such as the quantity of permissions or API calls. The encoder lowers the dimensionality to provide a representation of latent space(Xing et al., 2022). Common activation functions used are sigmoid and ReLU. Latent Space is a condensed depiction of the input characteristics. Finally, the decoder restores the original dimensions of the input from the latent space. Reconstruction loss between the initial input and the reconstructed result can be reduced by using Mean Squared Error (MSE). Unsupervised training of the AE using a dataset of dangerous and benign applications is required.

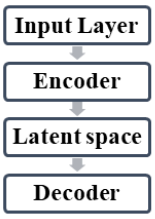

Figure 2 Architecture of AE

Following training, extract latent features from the input data using the AE's component. The original data is condensed and meaningfully represented by the latent characteristics (Bakr et al., 2023). Transfer the latent characteristics that were extracted to a classification model using TL models, such as MobileNetv3.

MobileNetV3

A CNN called MobileNetV3 is optimized for mobile phone CPUs by combining the NetAdapt algorithm with hardware-aware network architecture search (NAS), and then further enhanced by new architectural developments. Complementary search strategies, new effective nonlinearity versions that are appropriate for mobile environments and new effective network architecture are among the advancements. In the MBConv blocks, the network architecture makes use of squeeze-and-excitation modules and a hard swish activation.

A lightweight and effective CNN architecture, MobileNetv3 is intended for mobile and edge devices. When used in conjunction with feature extraction methods like AEs that deal with structured data like permissions, API requests, or dynamic behavior logs, it works well for classifying Android malware. The components of the MobileNetV3 includes

- The stem layer reduces spatial dimensionality by using stride 2 in the initial convolutional layer.
- Bottleneck blocks include Inverted Residual Blocks, which involve projecting the input tensor back after applying depthwise separable convolution. For channel-wise attention, SE (Squeeze-and-Excitation) blocks are optional.
- Head Layers: Fully linked classification layers, global average pooling, and final convolutional layers. The table 1 lists the layers in MobileNetV3 small architecture

Table 1 MobileNetV3 small Architecture

Layer	Size of input	Operator	Expansion	Output channels	SE	NL	Stride
Conv2D	224^224*3	3*3 Conv	-	16	No	h-swish	2
Bottleneck	112*112*16	3*3 DWConv	16	16	No	ReLU	1
Bottleneck	112*112*16	3*3 DWConv	64	24	No	ReLU	2
Bottleneck	56*56*24	3*3 DWConv	72	24	No	ReLU	1
Bottleneck	56*56*24	5*5 DWConv	72	40	Yes	ReLU	2
Bottleneck	28*28*40	5*5 DWConv	120	40	Yes	ReLU	1
Bottleneck	28*28*40	5*5 DWConv	120	40	Yes	ReLU	1
Bottleneck	28*28*40	3*3 DWConv	240	80	No	h-swish	2
Bottleneck	14*14*80	3*3 DWConv	200	80	No	h-swish	1
Bottleneck	14*14*80	3*3 DWConv	184	80	No	h-swish	1
Bottleneck	14*14*80	3*3 DWConv	184	80	No	h-swish	1
Bottleneck	14*14*80	3*3 DWConv	480	112	Yes	h-swish	1
Bottleneck	14*14*112	3*3 DWConv	672	112	Yes	h-swish	1
Bottleneck	14*14*112	5*5 DWConv	672	160	Yes	h-swish	2
Bottleneck	7*7*160	5*5 DWConv	960	160	Yes	h-swish	1
Bottleneck	7*7*160	5*5 DWConv	960	160	Yes	h-swish	1
Conv2D+Pool	7*7*160	1*1Conv	-	1280	No	h-swish	-
Fully Connected (FC)	1*1*1280	Fully Connected	-	#Classes	No	Softmax	-

Convolutional Layers (conv2D) perform simple convolution used to extract spatial characteristics. Separable Convolutions by Depth divide a typical convolution into two sections. Apply a single filter to each input channel for depthwise convolution. Then Pointwise convolution is to merge channels, use 1x1 convolutions. The residuals inverted project back to lower dimensions, perform procedures, and enlarge input channels. Blocks of Squeeze-and-Excitation (SE) reweight features according to global statistics to add channel focus. The ReLu activation function used in the layers for efficiency and h-swish provides computationally efficient approximation. The fully connected layers combine all the features for the final classification (Zhou et al., 2023)

RESULTS

The studies were conducted using python in the colab environment. A Windows 11 OS with a core i7 processor and 8 GB of RAM was utilized. The under-sampled Android malware detection dataset is used from Kaggle for the analysis.

Dataset Description

The data is necessary to conduct more thorough malware investigations. Furthermore, in order to classify malware, ML and DL methods need a significant quantity of training data. Acquiring the right datasets for research projects is essential. Malicious software known as mobile malware targets wirelessly equipped personal digital assistants (PDAs) and mobile phones by crashing the device and leaking or losing private data. It has become more challenging to guarantee wireless phones' and PDA networks' security and safety against hacking attempts in the form of viruses or other malware as they have risen in popularity and sophistication. This study uses Android malware detection dataset from the Kaggle repository. This dataset includes feature vectors of 215 properties that were taken from 15,036 programs, including 9,476 benign apps and 5,560 malicious apps from the Drebin project.

Comparison Analysis

A number of noteworthy studies have been carried out in the field of malware detection using ML models that contrast with the proposed study. The following are the methods used for comparison analysis.

Study A : Support Vector Regression + RBF + CNN

SVR for malware classification uses a Radial Basis Function (RBF) kernel. With 95.74% accuracy, 94.76% precision, 98.06% recall, and a 96.38% F1-score, our results show that the SVR-based model performs better than benchmark algorithms like SVM, Random Forest, and CNN(Aldhafferiet al.,2024).

Study B: Transfer Learning + GNN

Transfer learning with Graph Neural Networks(Raza et al., 2024)

Study C: Attention Mechanism + LightGBM

The most crucial characteristics are chosen using a feature selection technique based on the Attention mechanism(AM). The Android samples are then classified and the malware is identified using an improved Light Gradient Boosting Machine (LightGBM) classifier (Ghourabi, et al., 2024).

The metrics like accuracy, precision, recall and f1-score (Albi et al., 2023) were used for evaluating the proposed model with the existing models.

- **Accuracy**: A common statistic for assessing how successfully a detection system differentiates between dangerous and benign apps is the accuracy metric in Android malware detection. It is determined by dividing the total number of instances by the proportion of correctly categorized cases (malicious and benign). However, because of the usual class imbalance in datasets, accuracy alone might not be enough in virus detection settings. The formula is shown in equation(1)

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1) \text{ i.e } Accuracy = \frac{Correctly\ classified\ samples}{Total\ Samples}$$

- **Precision**: calculates the percentage of harmful applications that are actually malware. It is computed using the equation (2)

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

- **Recall (Sensitivity)**: evaluates the system's malware detection capabilities.

$$Recall = \frac{TP}{(TP+FN)} \quad (3)$$

- **F1-Score**: It balances the precision and recall

$$F1 - score = 2 * \frac{Precision * recall}{precision + recall} \quad (4)$$

Even while accuracy is a helpful starting point, assessing Android malware detection systems generally requires an amalgamation of measures, particularly when class imbalance is present, such as precision, recall, and F1-score.

Table 1 shows the comparison results of the proposed model with study A, B and C in terms of accuracy, precision, recall and F1-score. Table 2 shows the comparison results with regard to the error metrics like MAE and MSE. These comparison results highlight how well the suggested model detects Android malware.

Table 2 Comparison Analysis –with Feature Extraction

Methods/Metrics	Accuracy	Precision	Re-call	F1-Score
AutoEncoder+MobileNetV3	97.9	96.5	97.4	96.8
SVR+RBF+CNN - Study A	91.4	90.2	89.8	90.3
TL+GNN-Study B	95.2	94.6	94.2	94.5
AM+LightBGM- Study C	95.9	92.3	93.5	94.1

Table 1 shows the model performances of the proposed method and the contrasted method . The suggested approach performs best on all parameters, achieving the greatest F1-score (96.8%), accuracy (97.9%), precision (96.5%), and recall (97.4%).
This technique effectively integrates a strong lightweight classification model (MobileNetV3) with feature extraction (AutoEncoder). SVR + RBF + CNN (Study A) performs the least well out of all the approaches, with F1-score (90.3%), accuracy (91.4%), precision (90.2%), and recall (89.8%). The intricacy and ineffectiveness of merging SVR and CNN, particularly for high-dimensional data, is probably the limit.
GNN + Transfer Learning (Study B) works well, with a balanced F1-score of 94.5% and accuracy of 95.2% and it is appropriate for capturing structural relationships such as API call graphs since it makes use of pre-trained models and GNN for relational data. LightGBM + Attention Mechanism (Study C) achieves a little lower F1-score (94.1%) but a slightly greater accuracy (95.9%) than Study B. While LightGBM guarantees excellent and efficient categorization, attention processes aid in identifying important characteristics.

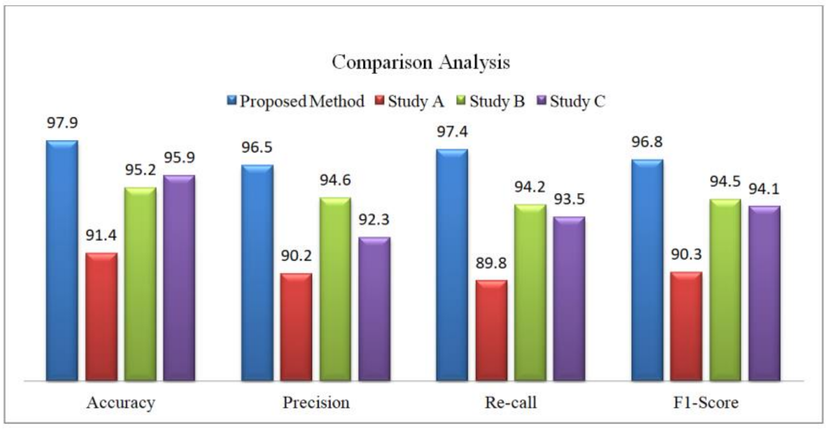

Figure 3. Metrics Vs Methods

From figure 3, it is evident that the proposed method has higher performance metrics and high accuracy and it makes effective use of compressed latent representations for categorization. This illustrates how MobileNetV3 and AE based feature extraction work better together to identify malware. It successfully detects malware with a high recall and precision. SVR classification and integrated feature extraction (CNN) result in moderate performance. It requires more computing power than simpler models. AM with LightGBM provides balanced performance metrics The best-performing model is AutoEncoder + MobileNetV3, demonstrating the effectiveness of feature extraction in conjunction with a small yet potent classifier. While they both perform well and are competitive, TL + GNN and Attention Mechanism + LightGBM have somewhat distinct advantages. Perhaps because SVR and CNN are inefficient when combined for this application, SVR + RBF + CNN perform worse. This study demonstrates the advantages of each method in detail and backs up the selection of AutoEncoder + MobileNetV3 for the best Android malware detection.
Metrics such as Mean Squared Error (MSE) and Mean Absolute Error (MAE) are used to compare the various models (AutoEncoder+MobileNetV3, SVR+RBF+CNN, TL+GNN, and AM+LightGBM) and the MSE and MAE values are listed in table 3. In some situations, Android malware detection can make use of metrics like MSE and

MAE, which are frequently used in regression tasks. This is particularly true when the detection model produces continuous probabilities or risk scores rather than binary classifications (malware or benign)(Alkahtani et al., 2022). These metrics are related to Android malware detection in the following ways:
MSE: The average squared difference between expected and actual values is measured by MSE.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \qquad (5)$$

In equation (5) $\hat{y}_i$: Predicted value, such as the likelihood that an application contains malware. y_i. The actual value, such as the ground truth label, where benign = 0 and malware = 1 and n is the overall quantity of samples.
MAE: The average absolute difference between expected and actual values is measured by MAE.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \qquad (6)$$

In equation (6) , $\hat{y}_i$ Predicted value and y_i is the actual value and n is the number of samples

Table 3 Comparison Analysis-Error Metrics

Methods/Metrics	MSE	MAE
AutoEncoder+MobileNetV3	0.002	0.004
SVR+RBF+CNN - Study A	0.011	0.013
TL+GNN-Study B	0.006	0.007
AM+LightBGM- Study C	0.005	0.004

From table 3 , it is clear that the AutoEncoder + MobileNetV3 is effective at reducing significant errors (MSE = 0.002). Moreover, variations in malware types, class balance, and dataset size can all affect MSE and MAE.

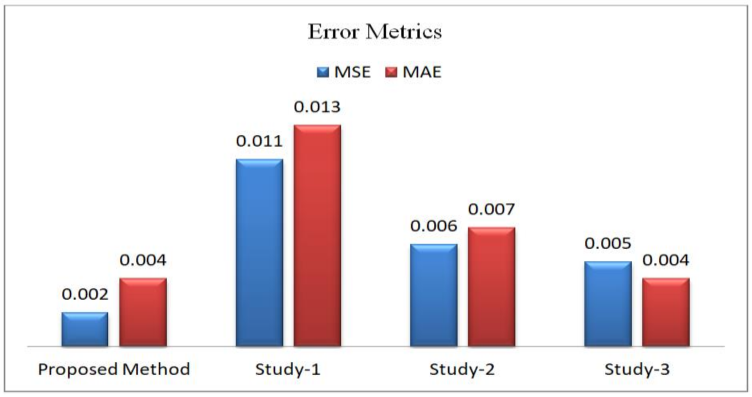

Figure 4 Methods Vs Error Metrics

From figure 4, it is found that the AutoEncoder + MobileNetV3 has the lowest error and the best performance in terms of both MSE and MAE. The most significant errors are seen in SVR + RBF + CNN (Study A), suggesting that it may not perform as well as the other models. Compared to AutoEncoder + MobileNetV3, TL + GNN (Study B) and AM + LightBGM (Study C) exhibit competitive performance, but with slightly higher errors. The best combination to minimize errors in Android malware detection is AutoEncoder + MobileNetV3, closely followed by AM + LightGBM. However, in addition to error metrics, the method's choice should take into account runtime efficiency, scalability, and interpretability.

Discussions

AE and MobileNetV3 together can offer a number of benefits, particularly for jobs involving feature extraction, dimensionality reduction, or picture reconstruction. In order to reduce dimensionality and let the model concentrate on the most significant aspects of the data, AE can learn the compressed form (latent space) of the input data.
With its efficient architecture, MobileNetV3 is a lightweight CNN that can extract features from photos without consuming a lot of processing power. MobileNetV3 may be used in situations where processing power is constrained because it was developed for devices that are mobile or embedded. The model may be both computationally economical and feature extraction-effective when used with an AE. Because the AE may lower the dimensionality of the data, downstream processes require fewer calculations.

AE models are often beneficial for data compression or denoising, which can improve MobileNetV3 performance by emphasizing on the important aspects and eliminating unnecessary or data that is noisy. The combined method is more resource-effective while maintaining high precision for real-time applications like anomaly detection or image classification.

AutoEncoders may identify hidden trends or characteristics in data without using any datasets with labels since they are frequently employed in unsupervised learning. The system gains from both the unsupervised representation learning of AEs and the potent TL capabilities of MobileNetV3, which is optimized for effective feature extraction. AE can assist in removing noise from the input data, strengthening it and enabling the MobileNetV3 to concentrate on the most significant patterns. Because of this, the combined method works well for jobs where preserving the level of accuracy of the input data is crucial, such as image denoising or reconstruction. Because the AE reduces the dimensionality, the model will concentrate on fewer but more significant characteristics, hence avoiding overfitting. Even with very little datasets, MobileNetV3's effective parameter use allows the model to perform well without overfitting. Effective feature extraction and dimensionality reduction make the AE + MobileNetV3 combination perfect for real-time, low-resource, scalable applications with relatively high precision.

Despite its lightweight architecture, MobileNetV3 may nevertheless contain a lot more parameters than more straightforward models. This could reduce its effectiveness when used in settings with very restricted resources (such as mobile devices with little memory and processing capacity). If the network is too complicated, AEs may have problems like overfitting, particularly if the design is not appropriately adjusted for the dataset. Like many deep neural networks,AEs and MobileNetV3 are DL models that can be difficult to understand. It might be hard to understand the reasoning behind the model's forecasts or representations, which can be detrimental in cases where explainability is crucial.

CONCLUSION

An integrated DL approach for Android malware detection is presented in this work, which combines TL for classification with AEs for feature extraction. High-dimensional, latent characteristics are extracted using AEs from Android app analysis data that is both static (related to the code) and dynamic (related to behavior). As a result, both benign and destructive activities may be efficiently captured and represented by the system. TL uses deep neural networks that have already been trained to categorize Android applications more precisely. TL lowers the necessary training time and increases detection accuracy by employing models that have already been trained on a huge dataset. Tested on dataset, the suggested framework demonstrated 97.9% accuracy with an MSE of 0.002 and MAE of 0.004. This illustrates how well the model can identify both safe and dangerous Android applications. An effective and precise malware detection system that can be incorporated into mobile security solutions is produced by combining AEs for feature extraction with TL for classification. It has the potential to be implemented in mobile security apps, antivirus programs, and app marketplaces. TL may be used on mobile devices or in resource-constrained cloud-based security systems since it cuts down on the amount of time and computing power needed for training. Future studies might investigate more sophisticated feature extraction methods to capture more intricate patterns over time in dynamic analysis, such as hybrid models that combine AEs with Recurrent Neural Networks (RNNs) for sequential data.

REFERENCES

[1] K. Liu, G. Zhang, X. Chen, Q. Liu, L. Peng and L. Yurui, "Android malware detection based on sensitive patterns", *Telecommun. Syst.*, vol. 82, no. 4, pp. 435-449, Apr. 2023.

[2] Joshi, Apoorv & Kumar, Sanjay. (2023). Stacking-based ensemble model for malware detection in android devices. International Journal of Information Technology. 15. 10.1007/s41870-023-01392-7.

[3] N. Firoz, M. S. Tahsin and M. Z. Hossain, "Autoencoder-Driven Anomaly Detection for Efficient Malware Identification," *2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)*, Coimbatore, India, 2024, pp. 1-8, doi: 10.1109/ICoICI62503.2024.10696300.

[4] Roy, S., Bhanja, S., & Das, A. (2023). AndyWar: an intelligent android malware detection using machine learning. Innovations in Systems and Software Engineering, 1-9.

[5] Manzil, H. H. R., & Naik, S. M. (2024). Detection approaches for android malware: Taxonomy and review analysis. Expert Systems with Applications, 238, 122255.

[6] Liu, Z., Wang, R., Japkowicz, N., Gomes, H. M., Peng, B., & Zhang, W. (2024). SeGDroid: An Android malware detection method based on sensitive function call graph learning. Expert Systems with Applications, 235, 121125.

[7] Kumar, S., Ahlawat, P., & Sahni, J. (2024). IOT malware detection using static and dynamic analysis techniques: A systematic literature review. Security and Privacy, 7(6), e444.

[8] Bean, J., & Torri, S. A. (2024, November). Survey of Malware Detection through Use of Neural Network Models. In 2024 International Conference on Cyber-Physical Social Intelligence (ICCSI) (pp. 1-7). IEEE.

[9] Aldhafferi, N. (2024). Android Malware Detection Using Support Vector Regression for Dynamic Feature Analysis. *Information*, *15*(10), 658. https://doi.org/10.3390/info15100658

[10] Revaldo, D. 2022. Available online: https://www.kaggle.com/datasets/dannyrevaldo/android-malware-detection-dataset (accessed on 7 June 2024).

[11] Raza, A., Qaisar, Z. H., Aslam, N., Faheem, M., Ashraf, M. W., & Chaudhry, M. N. (2024). TL-GNN: Android Malware Detection Using Transfer Learning. Applied AI Letters, 5(3), e94.

[12] Ullah, F., Ullah, S., Srivastava, G., Lin, J. C. W., & Zhao, Y. (2024). NMal-Droid: network-based android malware detection system using transfer learning and CNN-BiGRU ensemble. Wireless Networks, 30(6), 6177-6198.

[13] R. Shen, H. -j. Zhu, C. Li and H. -h. Wei, "GAResNet: A Transfer Learning based Framework for Android Malware Detection," *2023 IEEE International Conference on Knowledge Graph (ICKG)*, Shanghai, China, 2023, pp. 263-268, doi: 10.1109/ICKG59574.2023.00038.

[14] Tasyurek, M., & Arslan, R. S. (2023). Rt-droid: a novel approach for real-time android application analysis with transfer learning-based cnn models. Journal of Real-Time Image Processing, 20(3), 55

[15] Bostani, H., & Moonsamy, V. (2024). Evadedroid: A practical evasion attack on machine learning for black-box android malware detection. Computers & Security, 139, 103676.

[16] Ghourabi, A. (2024). An Attention-Based Approach to Enhance the Detection and Classification of Android Malware. Computers, Materials & Continua, 80(2).

[17] Bakır, H., & Bakır, R. (2023). DroidEncoder: Malware detection using auto-encoder based feature extractor and machine learning algorithms. Computers and Electrical Engineering, 110, 108804.

[18] Xing, X., Jin, X., Elahi, H., Jiang, H., & Wang, G. (2022). A malware detection approach using autoencoder in deep learning. IEEE Access, 10, 25696-25706.

[19] Zou, B., Cao, C., Tao, F., & Wang, L. (2022). IMCLNet: A lightweight deep neural network for Image-based Malware Classification. Journal of Information Security and Applications, 70, 103313.

[20] Albin Ahmed, A., Shaahid, A., Alnasser, F., Alfaddagh, S., Binagag, S., & Alqahtani, D. (2023). Android ransomware detection using supervised machine learning techniques based on traffic analysis. Sensors, 24(1), 189.

[21] Alkahtani, H., & Aldhyani, T. H. (2022). Artificial intelligence algorithms for malware detection in android-operated mobile devices. Sensors, 22(6), 2268.

[22] Atacak, İ., Kılıç, K., & Doğru, İ. A. (2022). Android malware detection using hybrid ANFIS architecture with low computational cost convolutional layers. PeerJ Computer Science, 8, e1092.