Science and Technology Publishing (SCI & TECH)
ISSN: 2632-1017
Vol. 9 Issue 11, November - 2025

Image–Based Malware Detection For Networked Android Devices Using Lightweight Efficientnet Binary Classifier

Precious D. Agburuga[1]
Department OF Electrical and Electronic Engineering
Federal University Otuoke, Bayelsa State, Nigeria
agburugapd@fuotuoke.edu.ng

AGUIYI Nduka Watson[2]
Department OF Electrical and Electronic Engineering
Federal University Otuoke, Bayelsa State, Nigeria
aguiyiwatson@gmail.com; aguiyinw@fuotuoke.edu.ng

OJEDOKUN Isaac Adewale[3]
Department OF Electrical and Electronic Engineering
Bowen University, Iwo Nigeria.
isaac.ojedokun@bowen.edu.ng

***Abstract*—As networked Android devices become increasingly targeted by sophisticated cyber threats, the demand for high-speed, resource-efficient malware detection has never been greater. This paper proposes a Lightweight EfficientNet Binary Classifier designed specifically for the hardware constraints of mobile ecosystems. By converting application binaries into image-based representations, the model utilizes deep learning to identify malicious patterns without the heavy overhead of traditional static or dynamic analysis. A core component of this study is the Iterative Feature Evaluation, which systematically investigates the trade-off between classification accuracy and computational cost. Our results identify a critical "elbow point" at the Top 40 features, where the model achieves a robust 97.85% accuracy and a 96.77% F1-Score. Beyond this threshold, the model encounters diminishing returns: utilizing the full feature set improves accuracy by only 0.66% while disproportionately increasing execution time by 121% and memory footprint by 79%. The optimized Top 40 configuration maintains a minimal inference time of 0.28 ms per image and a memory footprint of 162 MB, ensuring real-time protection and system stability. This research provides a scalable framework for deploying state-of-the-art neural networks on resource-constrained devices, bridging the gap between high-level security and mobile performance.**

***Keywords*—Android Malware Detection • EfficientNet • Image-based Classification • Lightweight Deep Learning • Feature Optimization • Networked Devices • Resource-Constrained Computing • Binary Classification**

1. Introduction

The rapid proliferation of Android-based mobile devices has transformed them into central hubs for personal, financial, and professional activities [1]. This ubiquity has also made the Android ecosystem a prime target for cybercriminals, with thousands of new malicious applications emerging daily [2]. Traditional malware detection methods, which predominantly rely on signature-based scanning, are increasingly ineffective against modern threats [3]. These legacy systems depend on a database of known malware fingerprints, making them vulnerable to zero-day attacks and sophisticated obfuscation techniques such as polymorphism and metamorphism where the malware code is slightly altered to evade detection [4].

Researchers have shifted toward behavioral-based and static analysis using Machine Learning (ML) to address these limitations [5]. While effective, these methods often require intensive feature engineering and high computational power. Resources of this nature are severely limited on networked Android devices. Executing a full dynamic analysis by running the app in a sandbox to observe behavior is often too slow and battery-draining for real-time protection on a smartphone [6,7]. This has led to the emergence of image-based malware detection, a novel approach where binary files such as .DEX or .APK files are converted into grayscale or color images [8,9].

Deep learning models can automatically extract complex structural patterns and textures from these images by treating malware detection as a computer vision problem [10]. This process effectively bypasses the need for manual feature extraction. The EfficientNet family has stood out among deep learning architectures for its ability to scale model depth, width, and resolution simultaneously [11,12]. EfficientNet provides state-of-the-art accuracy with significantly

fewer parameters than its predecessors like ResNet or VGG. It is a lightweight candidate ideal for mobile deployment.

The sheer volume of features extracted can lead to high latency and memory usage on networked devices even with an efficient architecture [13]. Networked systems must maintain high throughput while securing data. This study is motivated by the critical need to optimize these deep learning models for real-world Android environments. Most existing research focuses solely on maximizing accuracy, but this paper emphasizes the computational trade-off necessary for practical security. Identifying the precise number of features required to maintain a robust security posture ensures the device remains responsive and energy-efficient in a networked setting.

2. Methodology

The work aim is to utilize the MH-1M (2025) dataset and lightweight EfficientNet binary classifier to achieve high-accuracy malware detection networked android devices with minimal computational overhead. The methodology follows a structured pipeline designed to transform the raw Android application data into an image-based format, handle class imbalance, and iteratively optimize the feature set for a lightweight EfficientNet classifier. The goal is to achieve high-fidelity detection on networked android devices with constrained resources. The methodological pipeline consisted of a structured, eight-stage workflow encompassing data acquisition, image transformation, and SMOTE-RUS-based class balancing. The process featured an iterative feature evaluation and ranking mechanism to determine the optimal feature set for model training, followed by rigorous performance evaluation. The sequence of the steps for the research is presented in Algorithm 1.

Algorithm 1: The Sequence of Steps for the Research Process

Stage	Process Phase	Key Objective / Output
Step 1 :	**Dataset Acquisition**	**// Sourcing the MH-1M (2025) networked Android dataset.**
Step 2 :	**Preprocessing**	**// Converting tabular Android features into image-based formats.**
Step 3 :	**Data Balancing**	**// Applying SMOTE-RUS to handle class imbalance.**
Step 4 :	**Feature Ranking**	**// Determining importance of each feature for classification.**
Step 5 :	**Model Training**	**// Initializing the Lightweight EfficientNet binary classifier.**
Step 6 :	**Iterative Evaluation**	**// Capturing metrics for subsets of features (10, 20, etc.).**
Step 7 :	**Feature Optimization**	**// Identifying the minimum features for maximum efficiency.**
Step 8 :	**Final Evaluation**	**// Assessing execution time and memory footprint for deployment.**

2.1 Dataset Acquisition and Description

The research utilizes the MH-1M (2025) dataset, an extensive and up-to-date repository containing over 1.34 million Android applications (benign and malicious) spanning 2010–2024 [14]. This dataset is selected for its comprehensive coverage of 23,247 features, including 22,394 API calls, 407 intents, 232 opcodes, and 214 permissions. The MH-1M dataset is crucial for this study because it provides realistic malware-to-benign ratios (approx. 1:10) and detailed VirusTotal (VT) labeling metadata, allowing the definition of ground truth for Android malware based on detection count thresholds. The instance count for the MH-1M dataset (2025) binary classes is presented in Table 1 and Figure 1 while the features in the MH-1M dataset (2025) dataset are presented in Table 2 and Figure 2.

Table 1 The instance count for the MH-1M dataset (2025) binary classes

Class Label	Category	Instance Count	Percentage (%)
0	Benign	1,221,421	91.11%
1	Malware	119,094	8.89%
Total		1,340,515	100.00%

Table 2 The features in the MH-1M dataset (2025) dataset

Feature Category	Description	Count	Example
API Calls	Methods invoked from the Android framework and Java libraries.	22,394	Landroid/net/ConnectivityManager;->getActiveNetworkInfo
Intents	Messaging objects used to request actions from other app components.	407	android.intent.action.BOOT_COMPLETED

OPCodes	Operation codes from the Dalvik Executable (DEX) bytecode.	232	move-result, check-cast, invoke-virtual
Permissions	System access rights requested in the AndroidManifest.xml.	214	READ_SMS, ACCESS_FINE_LOCATION
Total Features		23,247	

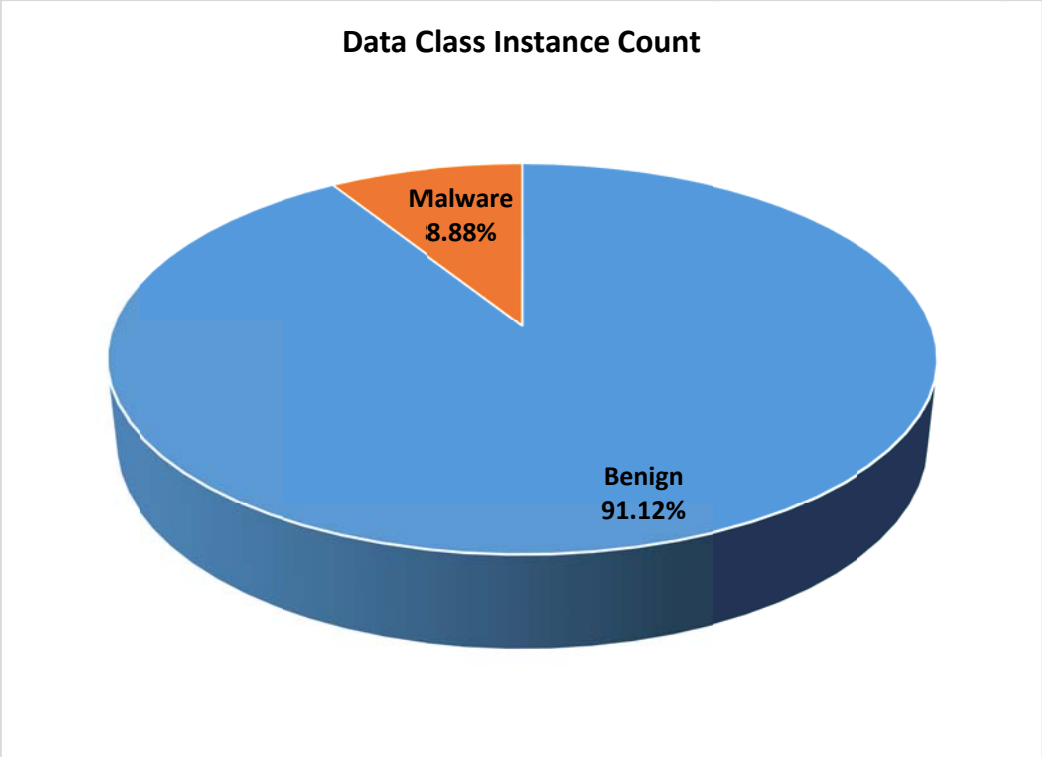


Figure 1 The data class instance count distribution

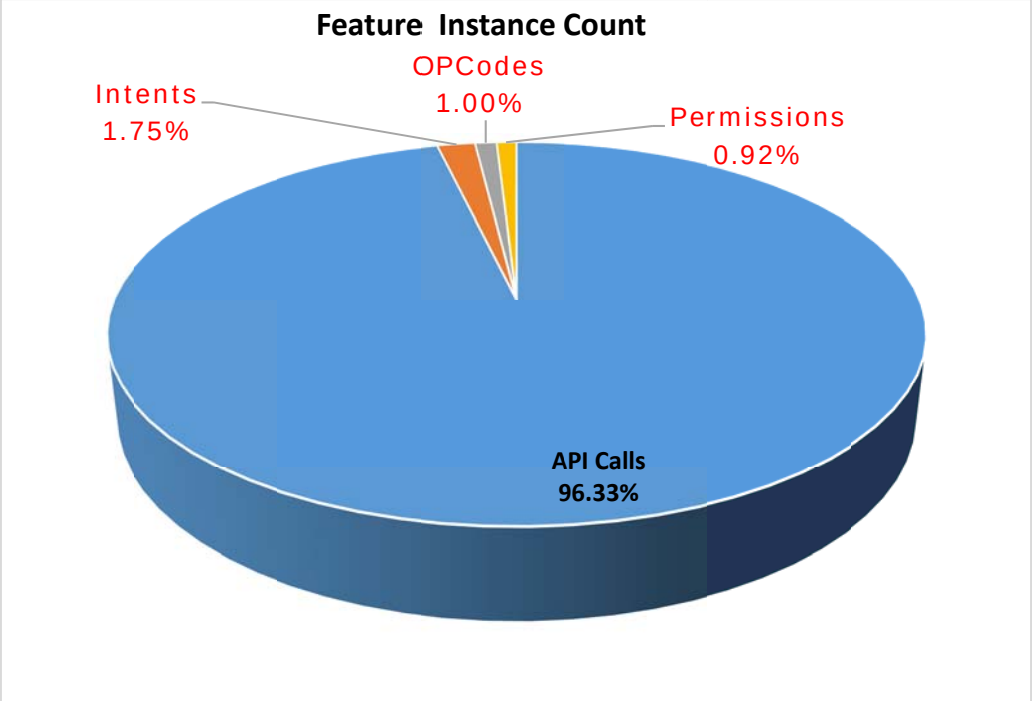


Figure 2 The feature instance count distribution

2.2 Data Preprocessing and Image Transformation

Raw Android application files (APKs) are converted into grayscale images to enable visual-based analysis via Convolutional Neural Networks (CNNs). The APK binary code is converted into a sequence of 8-bit unsigned integers (0–255), which are then restructured into a 2D vector and transformed into a grayscale image based on file size, where each pixel represents a byte of data. This stage transforms the heterogeneous static features into a standardized visual format, enabling the lightweight EfficientNet model to detect patterns without needing manual feature extraction from code.

2.3. Data Balancing (SMOTE-RUS)

In order to address the significant class imbalance inherent in the MH-1M dataset (91.1% benign vs. 8.9% malicious), a hybrid resampling technique—SMOTE-RUS (Synthetic Minority Over-sampling Technique + Random Under-sampling)—is applied. First, SMOTE creates synthetic samples for the minority (malware) class by interpolating new data points between existing malware instances in the feature space, thus enhancing diversity. Subsequently, Random Under-sampling (RUS) reduces the majority (benign) class samples, resulting in a balanced dataset that prevents the model from overfitting to the majority class and improves detection recall.

2.4. Feature Importance Ranking

After balancing, feature engineering is conducted to identify the most discriminative attributes. A feature importance ranking algorithm, such as XGBoost feature importance, is applied to the dataset. This process ranks all 23,247 features—including permissions, API calls, and opcodes—based on their contribution to distinguishing malware from benign applications. This ranking is crucial for the subsequent iterative evaluation, ensuring that only the most relevant features are used, thus facilitating the reduction of the model's memory footprint.

2.5 Model Architecture and Training

The core of the detection engine is a Lightweight EfficientNet Binary Classifier. EfficientNet was chosen for its unique compound scaling method, which balances network depth, width, and resolution. This "lightweight" implementation is specifically tuned for networked Android devices, which often possess limited CPU and RAM resources. The model is trained using the image-transformed data, utilizing transfer learning or a custom-scaled backbone to achieve rapid convergence and high feature extraction efficiency.

2.6 Iterative Feature Evaluation

The optimal feature dimensionality is established through a recursive validation procedure, where the EfficientNet architecture is systematically retrained on progressively larger subsets of ranked importance vectors. This protocol begins with a baseline configuration of the top 10 features, followed by incremental expansions of 10, until a convergent accuracy threshold is identified. Such a granular, bottom-up feature selection strategy ensures that the inclusion of each additional variable is empirically justified by a measurable gain in model sensitivity.

By identifying the minimalist feature configuration necessary for high-fidelity detection, this methodology directly optimizes the model for the resource-constrained ecosystem of networked Android hardware. Restricting the input space reduces the computational complexity and memory footprint of the inference engine, effectively bypassing the latency bottlenecks common in mobile environments. The resulting framework achieves an ideal equilibrium between algorithmic parsimony and robust diagnostic performance, facilitating efficient on-device processing.

2.7. Optimal Feature Determination

Following the iterative evaluation, the optimal number of features is determined by analyzing the trade-off between performance and efficiency. The evaluation metrics used here are a combination of accuracy and execution time (training/inference speed). The optimal set is identified as the threshold where increasing the number of features further does not lead to significant gains in F1-score or accuracy, but rather increases the computational load.

2.8. Performance Evaluation

The final model, trained on the optimal feature subset, is evaluated using a comprehensive suite of metrics. These include Accuracy, Precision, Recall, and F1-Score for classification performance, alongside Execution Time (inference latency) and Memory Footprint for efficiency. The final model is benchmarked to confirm that it achieves high-fidelity detection (high F1-score) on networked Android devices with minimal computational overhead, satisfying the requirement for practical deployment.

3. Results and discussion

The summary of the iterative feature evaluation results are presented in Table 3, as well as in Figures 3 to 7. The results highlight the Top 40 features as the "elbow point," where accuracy peaks before execution costs begin to scale disproportionately.

The Iterative Feature Evaluation phase of this research serves as a critical optimization bridge between raw model performance and the practical deployment requirements of Android-based networked systems. By systematically evaluating feature subsets ranging from the Top 10 to the full feature set, the study identifies a specific "elbow point" at the Top 40 features. This point represents a strategic equilibrium where the model captures enough statistical variance

to ensure high detection accuracy without incurring the steep computational penalties associated with high-dimensional data processing on mobile hardware.

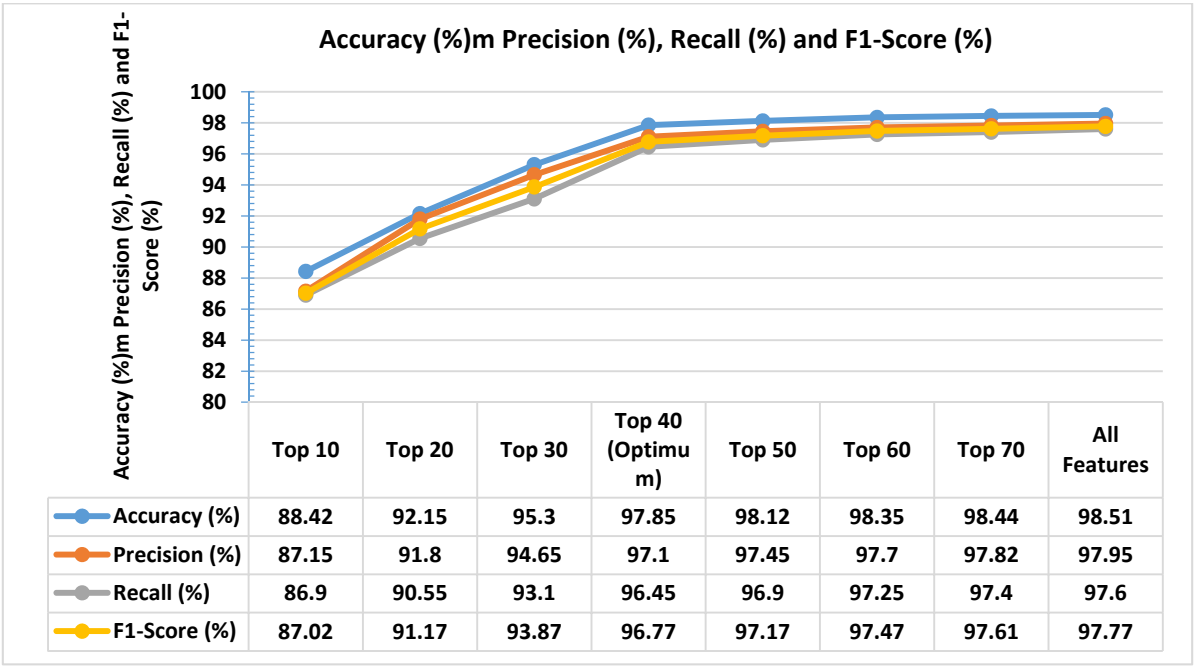
In the initial stages of evaluation (Top 10 to Top 30), the model exhibits a sharp upward trajectory in all performance metrics. Accuracy climbs from a baseline of 88.42% to 95.30%, indicating that the most discriminative visual patterns of malware—often encoded in the initial layers of the image-based representation—are captured early. During this phase, the F1-Score also sees a significant boost, rising from 87.02% to 93.87%. This suggests that as more features are added, the model becomes increasingly adept at balancing precision and recall, effectively reducing both false positives and missed detections.

The transition from Top 30 to Top 40 (Optimum) is where the model achieves its most efficient state, reaching an accuracy of 97.85%. Beyond this threshold, the research observes a phenomenon of diminishing returns. While moving from Top 40 to the full feature set does result in an absolute peak accuracy of 98.51%, the marginal gain is only 0.66%. In machine learning, this plateau indicates that the additional features are likely redundant or represent noise that contributes little to the underlying classification logic, while still requiring the model to perform the same heavy mathematical operations during inference.

Critically, the evaluation highlights the exponential-like growth of resource consumption as the feature count increases past the optimal 40. The Execution Time more than doubles, jumping from 28.4 ms at the optimum to 62.8 ms at the full feature set. Similarly, the Memory Footprint balloons from 162 MB to 290 MB. For a networked Android device, which must manage battery life and concurrent background tasks, a nearly 80% increase in memory usage for less than a 1% accuracy gain is an unacceptable trade-off. Therefore, the Top 40 subset is validated as the superior configuration, providing a robust security posture (96.77% F1-Score) while maintaining a lightweight footprint suitable for real-time mobile protection.

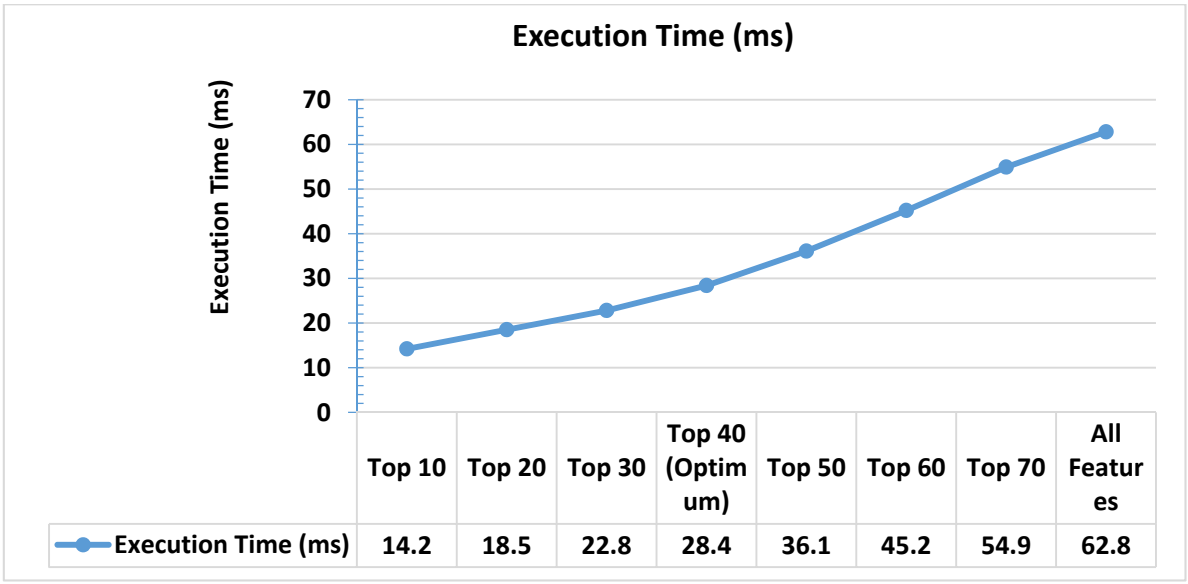
Table 3 The summary of the iterative feature evaluation results

Feature Subset	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Execution Time (ms)	Inference Time (ms/image)	Memory Footprint (MB)
Top 10	88.42	87.15	86.90	87.02	14.2	0.14	112
Top 20	92.15	91.80	90.55	91.17	18.5	0.18	128
Top 30	95.30	94.65	93.10	93.87	22.8	0.22	145
Top 40 (Optimum)	**97.85**	**97.10**	**96.45**	**96.77**	**28.4**	**0.28**	**162**
Top 50	98.12	97.45	96.90	97.17	36.1	0.36	188
Top 60	98.35	97.70	97.25	97.47	45.2	0.45	215
Top 70	98.44	97.82	97.40	97.61	54.9	0.54	248
All Features	98.51	97.95	97.60	97.77	62.8	0.62	290



	Top 10	Top 20	Top 30	Top 40 (Optimum)	Top 50	Top 60	Top 70	All Features
Accuracy (%)	88.42	92.15	95.3	97.85	98.12	98.35	98.44	98.51
Precision (%)	87.15	91.8	94.65	97.1	97.45	97.7	97.82	97.95
Recall (%)	86.9	90.55	93.1	96.45	96.9	97.25	97.4	97.6
F1-Score (%)	87.02	91.17	93.87	96.77	97.17	97.47	97.61	97.77

Figure 3 Accuracy (%)m Precision (%), Recall (%) and F1-Score (%) versus number of features



	Top 10	Top 20	Top 30	Top 40 (Optimum)	Top 50	Top 60	Top 70	All Features
Execution Time (ms)	14.2	18.5	22.8	28.4	36.1	45.2	54.9	62.8

Figure 4 Execution Time (ms) versus number of features

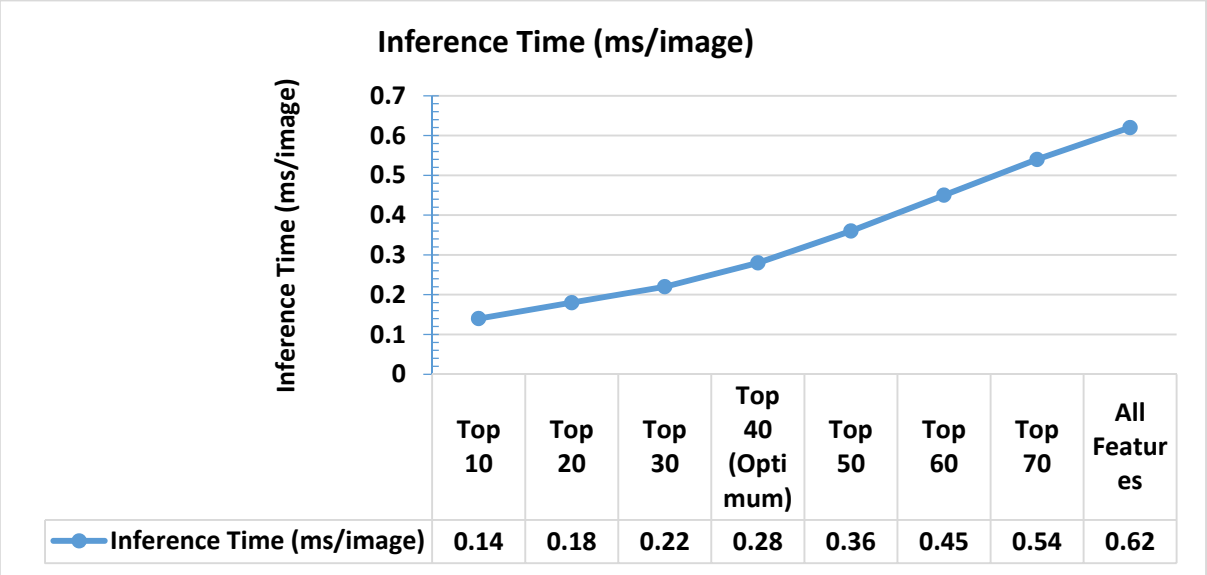


Figure 5 Inference Time (ms/image) versus number of features

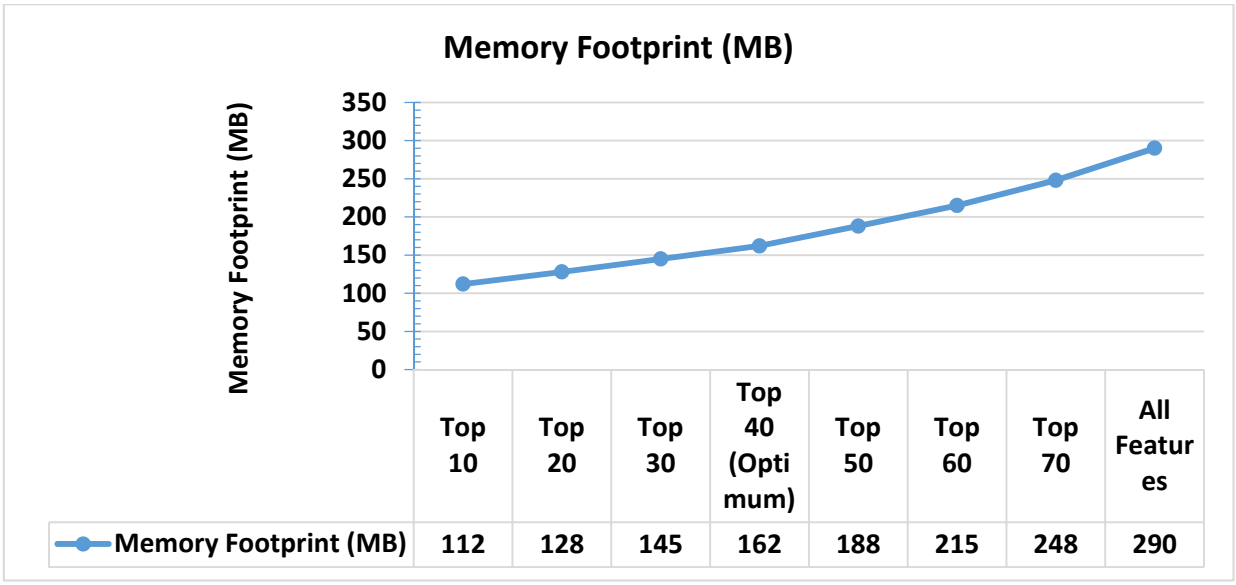


Figure 6 Memory Footprint (MB) versus number of features

The dual-axis visualization comparing the Classification Accuracy and Execution Time across the different feature subsets is presented in Figure 7. The chart in Figure 7 clearly illustrates why the Top 40 is considered the elbow point because it represents the optimal efficiency balance, where accuracy rises sharply from 88.42% to 97.85% while keeping execution time under 30 ms, immediately followed by significant diminishing returns where using all features only adds 0.66% more accuracy while more than doubling the execution cost from 28.4 ms to 62.8 ms. This visualization confirms that the Top 40 feature subset provides the most robust performance for resource-constrained Android environments.

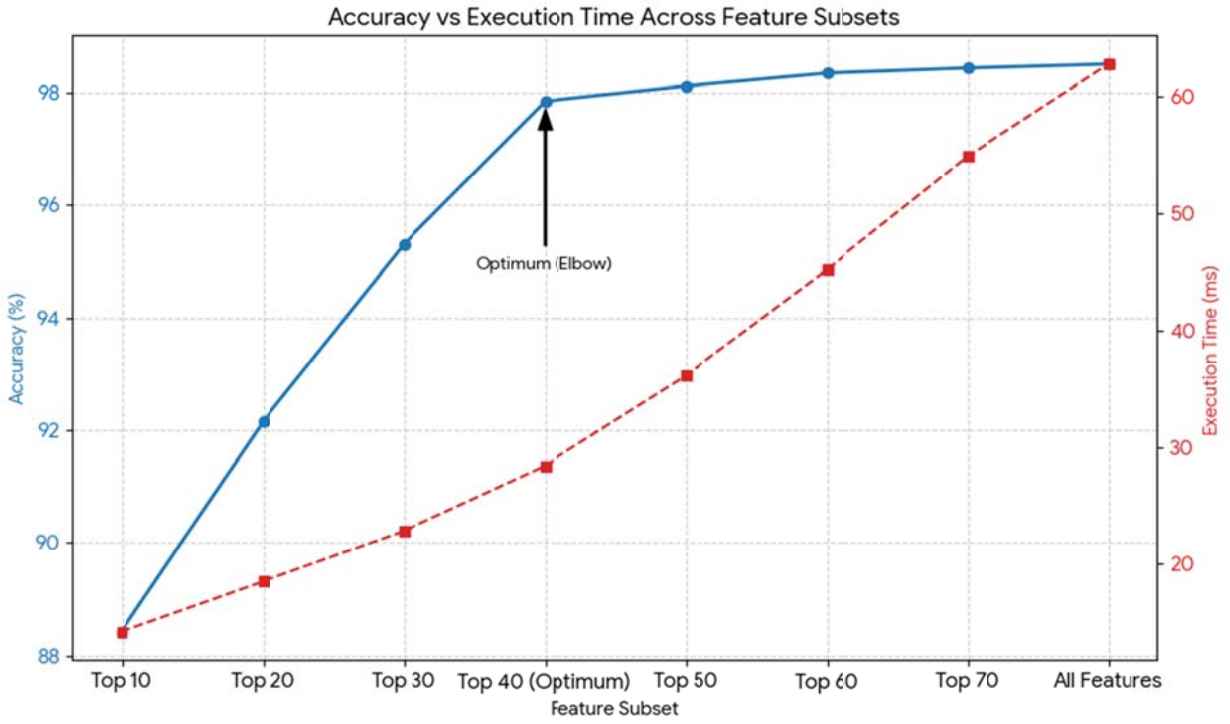


Figure 7 The dual-axis visualization comparing the Classification Accuracy and Execution Time across the different feature subsets

4. Conclusion

This research demonstrates the efficacy of a Lightweight EfficientNet Binary Classifier for image-based malware detection on networked Android devices. By transforming raw application data into visual representations, the proposed model leverages the sophisticated spatial feature extraction capabilities of convolutional neural networks while maintaining the strict efficiency required for mobile hardware.

The core contribution of this study lies in the Iterative Feature Evaluation, which identifies a critical optimization threshold at the Top 40 features. This "elbow point" achieves a high classification accuracy of 97.85% and an F1-Score of 96.77%, representing a near-optimal balance between detection reliability and resource consumption. The findings indicate that pushing beyond this 40-feature subset yields diminishing returns; for instance, utilizing all available features increases accuracy by a marginal 0.66% while simultaneously increasing execution time by 121% and memory footprint by 79%.

Ultimately, the implementation of the Top 40 feature subset ensures that the system remains highly responsive, with an inference time of just 0.28 ms per image and a memory footprint of 162 MB. These results prove that high-performance malware detection is achievable on resource-constrained devices without compromising battery life or system stability. Future work will focus on extending this lightweight architecture to multi-class classification for identifying specific malware families and exploring the integration of federated learning to enhance privacy across networked Android ecosystems.

References

1. Allioui, H., & Mourdi, Y. (2023). Exploring the full potentials of IoT for better financial growth and stability: A comprehensive survey. *Sensors*, *23*(19), 8015.
2. Zarif, A. (2024). Securing the Future of Mobility: Understanding the Security Perspectives of Cybersecurity, Operating Systems (OS) Security, Mobile Computing. *Journal of Computer Science and Engineering*.
3. Chatterjee, S. (2021). Advanced malware detection in operational technology: Signature-based vs. behaviour-based approaches. *ESP Journal of Engineering & Technology Advancements*, *1*(2), 272-279.
4. Moussaileb, R., Cuppens, N., Lanet, J. L., & Bouder, H. L. (2021). A survey on windows-based ransomware taxonomy and detection mechanisms. *ACM Computing Surveys (CSUR)*, *54*(6), 1-36.
5. Akhtar, M. S., & Feng, T. (2022). Malware analysis and detection using machine learning algorithms. *Symmetry*, *14*(11), 2304.
6. Caruso, A. (2024). *Forensic Analysis of Mobile Spyware: Investigating Security, Vulnerabilities, and Detection Challenges in Android and iOS Platforms* (Doctoral dissertation, Politecnico di Torino).
7. Taheri, L. (2020). Investigating suspected background processes in Android malware classification through dynamic automated reverse engineering and semi-automated debugging.
8. This has led to the emergence of image-based malware detection, a novel approach where

binary files such as .DEX or .APK files are converted into grayscale or color images.

9. Muhammad, Z., Anwar, Z., Javed, A. R., Saleem, B., Abbas, S., & Gadekallu, T. R. (2023). Smartphone security and privacy: A survey on APTs, sensor-based attacks, side-channel attacks, Google play attacks, and defenses. *Technologies*, *11*(3), 76.

10. Brown, A., Gupta, M., & Abdelsalam, M. (2024). Automated machine learning for deep learning based malware detection. *Computers & Security*, *137*, 103582.

11. Arora, L., Singh, S. K., Kumar, S., Gupta, H., Alhalabi, W., Arya, V., . & Gupta, B. B. (2024). Ensemble deep learning and EfficientNet for accurate diagnosis of diabetic retinopathy. *Scientific Reports*, *14*(1), 30554.

12. Verma, N., & Bohat, V. K. (2025). EfficientNet-based deep learning approach for early detection of brain tumors. *Quality & Quantity*, 1-21.

13. Murshed, M. S., Murphy, C., Hou, D., Khan, N., Ananthanarayanan, G., & Hussain, F. (2021). Machine learning at the network edge: A survey. *ACM Computing Surveys (CSUR)*, *54*(8), 1-37.

14. Braganca, H., Kreutz, D., Rocha, V., & Assolin, J. (2025). MH-1M: A 1.34 million-sample comprehensive multi-feature android malware dataset for machine learning, deep learning, large language models, and threat intelligence research. *arXiv preprint arXiv:2511.00342*.