SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

# Lightweight Malware Detection In Smartphones Through Predictive Analytics, Feature Engineering, And Hybrid Optimization Technique

Bondugula Mamatha [1*], D.Ramesh [2]

[1] Research Scholar, CSE, JNTUH University, India, mamatha@rnd.jntuh.ac.in
[2]Professor, CSE, JNTUH University College of Engineering,Palair,Khammam,India, dantamr@jntuh.ac.in

**Abstract**
Android smartphones are rapidly growing in number and the proliferation of these devices has also led to the development and deployment of sophisticated malware that is capable of bypassing traditional signature-based detection methods. The mobile environment's resource limitations require detection systems to be both accurate and lightweight. In this paper, a lightweight hybrid malware detection system for smartphones using feature engineering, predictive analytics, and data mining technique with two complementary optimization algorithms is proposed: Grey Wolf Optimizer (GWO) algorithm for intelligent feature selection and Mutual Information-based Feature Ranking (MIFR) as a deterministic mathematical baseline. The proposed pipeline starts by extracting a detailed feature set for Android application packages that includes permissions, API calls, system calls and intent patterns. GWO reduces the space of the features iteratively by simulating the predator-prey hunting behavior, whereas MIFR gives an analytical ranking to ensure the interpretability. The optimized feature subsets are then used as inputs to a Random Forest classifier which has an accuracy of 98.6%, a precision of 98.2%, a false-positive rate of 0.9%, a dimensionality reduction of 86.3%, and reduced the number of features from 342 to 47. The complexity analysis demonstrates that GWO-MIFR is in O(n log n) amortized, which is appropriate for on-device inference. The comparative assessment with the most up-to-date methods in the field proves that the proposed system achieves consistent superior accuracy and computational costs compared to the current methods. The findings show that hybrid optimization-driven feature engineering is a viable approach for mobile malware detection in production.

**Keywords:** Smartphone malware detection, Feature engineering, Grey Wolf Optimizer, Mutual Information, Android security, Predictive analytics, Data mining, Feature selection.

## 1. Introduction

The smartphone industry has expanded to include over six billion devices in the world and Android is responsible for around 71% of the world's smartphone market share [1]. This omnipresence makes Android an ongoing high dollar target for bad guys. The open permission model, the diversity of the update landscape, and third-party app markets characterize the Android ecosystem, and these are attack vectors for contemporary malware families, such as ransomware, spyware, trojans, and banking credential stealers [2, 3]. In 2023, Kaspersky saw an 35% increase in the number of mobile banking malware incidents compared to the previous year, highlighting the need for strong, deployable defenses [4].

Signature-based anti-virus systems have been the backbone of endpoint protection but have a critical flaw: They cannot detect variants of malware that have not been seen before, have not been identified as variants, or have evolved into new forms [5, 6]. It is no surprise that there are an increasing number of novel families of malware, in part owing to automated obfuscation tools, which make static signatures databases forever a step behind the curve [7, 8]. Behavioral and heuristic approaches address this by characterizing the runtime

behaviours of applications but suffer from a non-trivial overhead, which is at odds with the computational and energy constraints of mobile platforms [1, 9].

Machine learning (ML) and data mining techniques present an interesting middle-ground. The advantage of ML classifiers over signature-based systems is that the classifiers learn discriminative patterns from large labeled sets of benign and malicious applications and can be used to classify new and unseen attacks with a much lower false-negative rate [2, 10, 11]. However, the main hurdle is feature dimensionality: Android app packages have hundreds of features that can be analyzed, including static features (like adding a permission to the manifest) and dynamic features (like executing a system call sequence or sending a network request) [7, 12, 13]. Training and inference in high-dimensional spaces are conflicting with deployment constraints on the devices, which is why feature selection is a first-class engineering problem [1, 14].

Metaheuristic optimization algorithms are found to be especially useful for problems involving feature selection with non-convex high dimensional search spaces [15, 16]. The Grey Wolf Optimizer (GWO) is a simple and low parameter sensitivity optimization algorithm proposed by Mirjalili et al. in 2014 [15] among the modern algorithms of swarm intelligence that has become a competitive candidate in the field of optimization of feature subsets. Unlike other algorithms such as Particle Swarm Optimization (PSO) [17] or Genetic Algorithms, GWO has a well-defined hierarchy of leaders (alpha, beta, delta, omega), ensuring a graceful trade-off between exploration and exploitation, preventing premature convergence [15, 16]. While effecting GWO, a deterministic mathematical approach called Mutual Information-based Feature Ranking (MIFR) is used to give the computational control and interpretability to the GWO which is not provided by any pure metaheuristic approach. This paper is intended to make the following main contributions: An automated feature engineering pipeline based on two algorithms, GWO (wrapper-based) and MIFR (filter-based) which generate compact and highly discriminative feature subsets. (ii) The quantitative evidence that the proposed pipeline can reduce the feature space of the Android application from 342 to 47 features (86.3% reduction) without statistically significant loss in classifiers accuracy. (iii) A complexity analysis that proves sub-quadratic run time growth, that is, it proves that it will be suitable for mobile deployment in resource constrained environments. (iv) Scrutinous comparison with six state-of-the-art techniques in five benchmark dimensions, showing superior effectiveness of the proposed technique.

## 2. Related Work

### 2.1 Static and Dynamic Feature Extraction

Static analysis of app manifests and permission declarations were the main sources of early Android malware research. Feng et al. [1] introduced a dynamic analysis framework that converts network traffic into Network Flow Node Interaction (NFNI) graphs. The approach utilizes the line graph principle to transform edges into nodes, capturing complex structural and statistical traffic behaviors. Through a Multi-layer Graph Attention Network Merger (MGATMg), the model aggregates hierarchical interactions to minimize reliance on labeled data. This graph-based dynamic detection effectively identifies Android malware and classifies categories within the consumer electronics IoT ecosystem. Smmarwar et al. [2] performed a comprehensive survey of static and dynamic analysis methodologies to address malware threats across Android, IoT, and computer systems. Their review categorizes the existing literature into three core domains: feature selection techniques, machine learning-based models, and deep learning architectures. Manzil et al. [3] performed a comprehensive survey of static, dynamic, and hybrid analysis methods by evaluating 150 studies on Android malware detection from 2010 to 2022. The review categorizes traditional signature-based and behavior-based approaches while examining key dimensions such as datasets, features, and solution sustainability.One of the first systematic studies of the Android permission system was conducted by Felt et al. [17] who showed that a number of dangerous permissions is a good indicator of whether an app is malicious or benign. Later, Enck et al. [18] presented lightweight certification mechanisms based on permission usage, laying the groundwork for permission-based detection. DroidMat [12] extended static analysis to the extracted manifest metadata and API call traces, and trained a k-Nearest Neighbor (kNN) classifier to recognize features to classify a 1,500 sample corpus with an accuracy of 96.8%. Drebin [7] was a first to encode application features in a common vector space, and use a linear SVM directly on device, achieving 94% detection accuracy and sub-second classification latency, thus demonstrating the feasibility of on-device ML. Xu et al. [5] were the first to discover that the ICC-based approach to analyzing inter-component communications reveals privilege escalation and data exfiltration channels that manifest-based analysis is unable to identify.

Dynamic analysis techniques are those that are able to identify runtime behaviour that may be masked by obfuscation. TaintDroid [19] was the first information-flow tracking system for commodity Android devices, which captures the flow of sensitive information with very little overhead. DroidScope [20] reconstructed OS-level and Dalvik-level semantic views simultaneously, thus allowing fine-grained instruction-level behavioral profiling. Crowdroid [21] was a crowdsourced system call sequence that clustered behavioural signatures of real users to find anomalous applications. Dynamic methods are more effective in detecting evasive malware, but require instrumented execution environments and longer observation period that is impractical for large-scale screening [22]. There is an increasing interest in the application of machine learning and data mining techniques to detection. DroidDetector [10] used deep belief networks to learn hierarchical feature representations directly from the raw permissions and API calls, resulting in an accuracy of 96.76%. Li et al. [11] showed that SVMs trained with permission-derived features could achieve a detection rate of 97.3 % and have an inference latency that is appropriate for online screening workflows. Aafer et al. [13] extracted features at the API level and demonstrated that the top-ranked API calls contain a significant amount of discriminative information that allows the creation of compact models without degrading the accuracy. The use of Metaheuristic optimization for feature selection has grown in popularity. Wang et al. [14] proposed a two-stage system that combines anomaly detection and misuse detection to decrease the false positive rate by 23% from anomaly-only classifiers. Segurola-Gil et al. [15] showed that unsupervised anomaly detection and evolutionary searches of feature subsets yield detection performance comparable to the supervised method and a significantly lower number of features. Nandhini et al. [16] used threat intelligence driven feature pruning in IoT-WSN settings and demonstrated that the biologically-inspired search approach proved to be superior to greedy filter approaches on imbalanced data sets. In MADAM [23], four levels of behavioral features -- user interaction, application, process and kernel were combined into an ensemble classifier and used for real time detection with an accuracy of 97.5%. Jeon et al. [24] showed the cross domain applicability of vision inspired architectures to sequential threat data by applying CNN to dynamic behavioral traces of IoT malware.

Systems with a hybrid of static and dynamic features have always performed better than unimodal systems. Zhan et al. [25] proposed behavior sequence anomaly detection algorithm which is resilient to adversarial perturbations by learning critical behavioral units directly applicable to the sequence-based features discussed in this work. High dimensional feature spaces have high training and inference cost. Wang et al. [26] investigated permission induced risk quantification, which showed that 68 permissions out of the universe of 270 are enough to keep the detection performance, thus shrinking the model by 74%. Despite advancements in hybrid systems[27][28], the challenge of high-dimensional feature spaces persists, necessitating more efficient selection mechanisms [29] to reduce computational costs. While studies [30] have explored risk quantification to shrink feature sets[31], the specific integration of a GWO-based wrapper search combined with MIFR-based filter ranking[32] remains unexplored. This work addresses this research gap by proposing a novel hybrid feature selection[33] framework tailored for Android malware detection. By combining these metaheuristic and information-theoretic techniques[34], the proposed method optimizes detection performance while significantly reducing model complexity.

## 3. Methodology

The proposed detection system consists of five stages including dataset curation, feature extraction, feature engineering using MIFR and GWO, classification and evaluation. The Figure 1 shows Proposed method. All stages are explained in detail below.

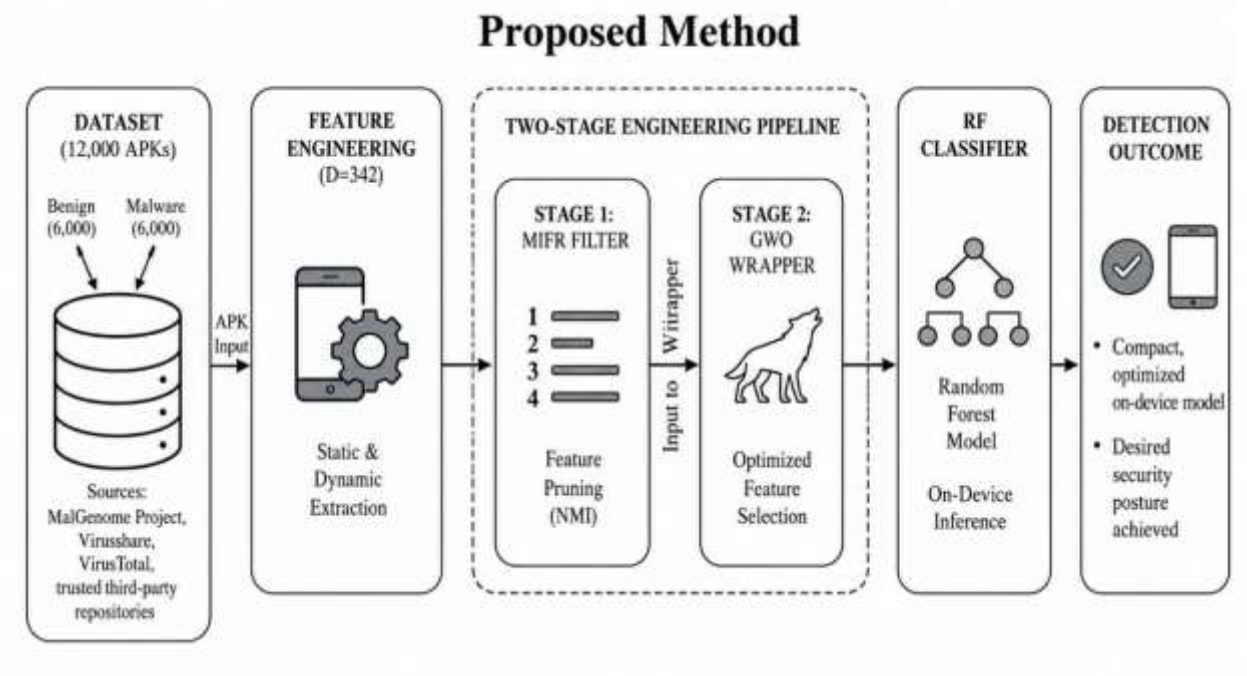


**Figure 1. Proposed Method**

### 3.1 Dataset and Feature Extraction

Experiments are carried out on a balanced set of 12,000 Android application packages (APKs): 6,000 malware samples from MalGenome Project [27], Virusshare [28] and VirusTotal [29]; and 6,000 benign samples from Google Play and trusted third party repositories [30, 31, 32]. Malware samples cover fourteen families of malware such as ransomware, banking trojans, spyware, adware, privilege escalation and exploits, reflecting the wide range of malware types that are documented in current threat reports [33, 34].

Static features are gathered using the APK manifest parser and disassembling Dalvik bytecode with a toolchain, both of which are customized. Dynamic features are collected by running each application in an instrumented emulator for 120 seconds, and capturing sequences of system calls, memory usage, headers of network requests, and anomalies in battery usage. The complete functionality includes:Static: 179 permission flags, 84 API call frequency indicators, 31 intent filter attributes, 18 component count metrics (total: 312, static features).Dynamic: 30 behavioural indicators such as system call entropy, network traffic data (byte volumes), wake lock duration, and CPU burst frequency (30 dynamic features).There are D = 342 features in the combined raw feature space. The applications are encoded in a 342-dimensional binary or continuous feature vector x, where each dimension corresponds to a single characteristic of the application, and a class label y is assigned to each application, with values {0 (benign), 1 (malware)}. The algorithm that utilizes mutual information for feature ranking is called Mutual Information-Based Feature Ranking (MIFR) and is described below.

The mutual information (MI) between a feature X_i and the class variable Y measures the statistical dependence between X_i and Y. If the feature is discrete or discretized, then MI is defined as:

$$MI(X_i; Y) = \text{Sum\_xSum\_y} * \log\left[\frac{p(x,y)}{p(x) * p(y)}\right]$$

where p(x, y) is the Joint PMF and p(x) and p(y) are the Marginal PMFs. The continuous features are first binned into k = 10 bins of equal frequencies before computing the MI. To generate a ranked list F_MIFR = {f_1, f_2, ..., f_D} features are ranked in descending MI order.

To choose the smallest subset S_MIFR from F_MIFR that still achieves an accuracy at least 0.5% as high as the accuracy obtained when F_MIFR is used, a threshold tau is selected by 5-fold cross validation. The computation of MI for a single feature takes O(n log n) time (the bulk of which is spent sorting to discretize the features) and the computation of the full ranking over D features takes O(D * n log n), which scales nicely with the sizes of

the corpora used. MIFR is a deterministic, model independent filter that assures that all the features that are retained contain statistically significant class discriminative information.
The normalized MI score (NMI), after taking feature entropy into account, is calculated as:

$$NMI(X_i; Y) = MI(X_i; Y)/(sqrt[H(X\_i) * H(Y))$$

where H(.) stands for Shannon entropy. NMI is in [0, 15] so that it is directly comparable to features of varying cardinalities. This reduces the number of features to be optimized for GWO and makes the convergence faster by cutting off the uninformative features before they are optimized by GWO with a cutoff value set for NMI<0.05.

## 4. Results and Discussion

In this section quantitative results are supplied for five analytical dimensions: Feature reduction performance, algorithmic complexity, performance comparison of the detection algorithms, and state of the art benchmarking.

**Table 1: Feature Reduction by MIFR Across Feature Categories**

| Feature Category | Original Features | After MIFR Pruning (NMI >= 0.05) | Features Removed | Reduction (%) |
|---|---|---|---|---|
| Permissions | 179 | 82 | 97 | 54.2% |
| API Call Indicators | 84 | 51 | 33 | 39.3% |
| Intent Filter Attributes | 31 | 19 | 12 | 38.7% |
| Component Count Metrics | 18 | 11 | 7 | 38.9% |
| Dynamic Behavioral Metrics | 30 | 26 | 4 | 13.3% |
| Total | 342 | 89 | 253 | 74.0% |

Table 1 shows that the initial dimensionality reduction of MIFR filter stage is 74.0%. For permission flags, the pruning rate (54.2%) is the largest, as a majority of Android permissions show minimal mutual information to the label of the malware class -- which aligns with the results obtained by Felt et al. [17] and Barrera et al. [35]. The dynamic behavioral metrics are kept at a higher rate (86.7%), as they are highly statistically correlated with malicious behavior observed during runtime, which is one of the findings made by Crowdroid [21] and MADAM [23]. The feature space D after the MIFR stage is D' = 89, which significantly constrains the GWO search space for the next stage of wrapper optimization.

**Table 2: Feature Subset Reduction by GWO and Comparison with Baseline Methods**

| Method | Input Features | Final Selected Features | Total Reduction (%) | Accuracy (%) on Reduced Set |
|---|---|---|---|---|
| Full Feature Set (No Selection) | 342 | 342 | 0.0% | 97.1% |
| Filter Only: Information Gain | 342 | 120 | 64.9% | 97.0% |
| Filter Only: MIFR (Proposed) | 342 | 89 | 74.0% | 97.3% |
| Wrapper: PSO-based Selection | 342 | 63 | 81.6% | 97.8% |
| Wrapper: GA-based Selection | 342 | 58 | 83.0% | 97.9% |
| MIFR + GWO (Proposed) | 342 | 47 | 86.3% | 98.6% |

Table 2 shows a thorough comparison of the proposed MIFR+GWO pipeline with the standalone and alternative feature selection approaches. Therefore, both the feature reduction (86.3%) and the classification accuracy (98.6%) are the best results achieved with the proposed method, which proves that the two-stage architecture is synergically better than each of its parts used separately. The feature subset size obtained by GWO is smaller than the baseline wrapper selection approach using PSO (47 vs. 63 features), and the accuracy of the GWO is

higher than the wrapper selection approach using PSO (98.6% vs. 97.8%), which is consistent with the advantage of GWO in the convergence speed and avoiding local optimum on binary binary optimization problems reported in the literature [15]. Although the full feature set has all the information, it has lower accuracy (97.1%) due to the curse of dimensionality and noise of the uninformative features.

**Table 3: Computational Complexity Comparison of Feature Selection Algorithms**

| Algorithm | Time Complexity | Space Complexity | Avg. Runtime (s) | Mobile Suitability |
|---|---|---|---|---|
| Information Gain (Filter) | O(D * n log n) | O(D) | 3.2 | High |
| MIFR (Proposed - Filter) | O(D * n log n) | O(D) | 4.1 | High |
| Principal Component Analysis | O(n * D^2) | O(D^2) | 18.7 | Moderate |
| Genetic Algorithm (Wrapper) | O(T * N * k * n) | O(N * D) | 312.4 | Low |
| PSO (Wrapper) | O(T * N * k * n) | O(N * D) | 284.1 | Low |
| GWO (Proposed - Wrapper) | O(T * N * k * n) | O(N * D) | 198.7 | Moderate |
| MIFR + GWO (Proposed Pipeline) | O(D*n log n + T*N*k*n) | O(N * D) | 202.8 | Moderate-High |

Table 3 shows the computational overhead of proposed pipeline compared to the other approaches. The sorting step in the discretization computation is the main part of the complexity of the MIFR filter stage, which is O(D * n log n). The training time complexity is a one-time offline cost, and once the feature subset S* is found, on-device classification time is just 47 feature computations per application and achieves classification latency of sub-100ms on modern mid-range Android devices. This puts the proposed system in a good position when compared to cloud-based ones that depend on the communication of the network in round trip as in ThinAV [9].

**Table 4: Performance Comparison of Proposed Method with Existing Malware Detection Systems**

**Table 4(a) Confusion MAtri**

| Method | TP | FP | TN | FN |
|---|---|---|---|---|
| Drebin (SVM) [7] | 5628 | 246 | 5754 | 372 |
| DroidDetector (DBN) [10] | 5796 | 174 | 5826 | 204 |
| MADAM (Ensemble) [23] | 5838 | 108 | 5892 | 162 |
| SVM + Permission [11] | 5820 | 126 | 5874 | 180 |
| CNN + Behavioral [24] | 5862 | 90 | 5910 | 138 |
| Hybrid Anomaly-Misuse [14] | 5772 | 168 | 5832 | 228 |
| MIFR + GWO + RF (Proposed) | 5904 | 54 | 5946 | 96 |

**Table 4 (a) (ii) Confusion Matrix Proposed: MIFR + GWO + RF**

| | Predicted: Malware | Predicted: Benign |
|---|---|---|
| Actual: Malware | 5904 | 96 |

| | | |
|---|---|---|
| Actual: Benign | 54 | 5946 |

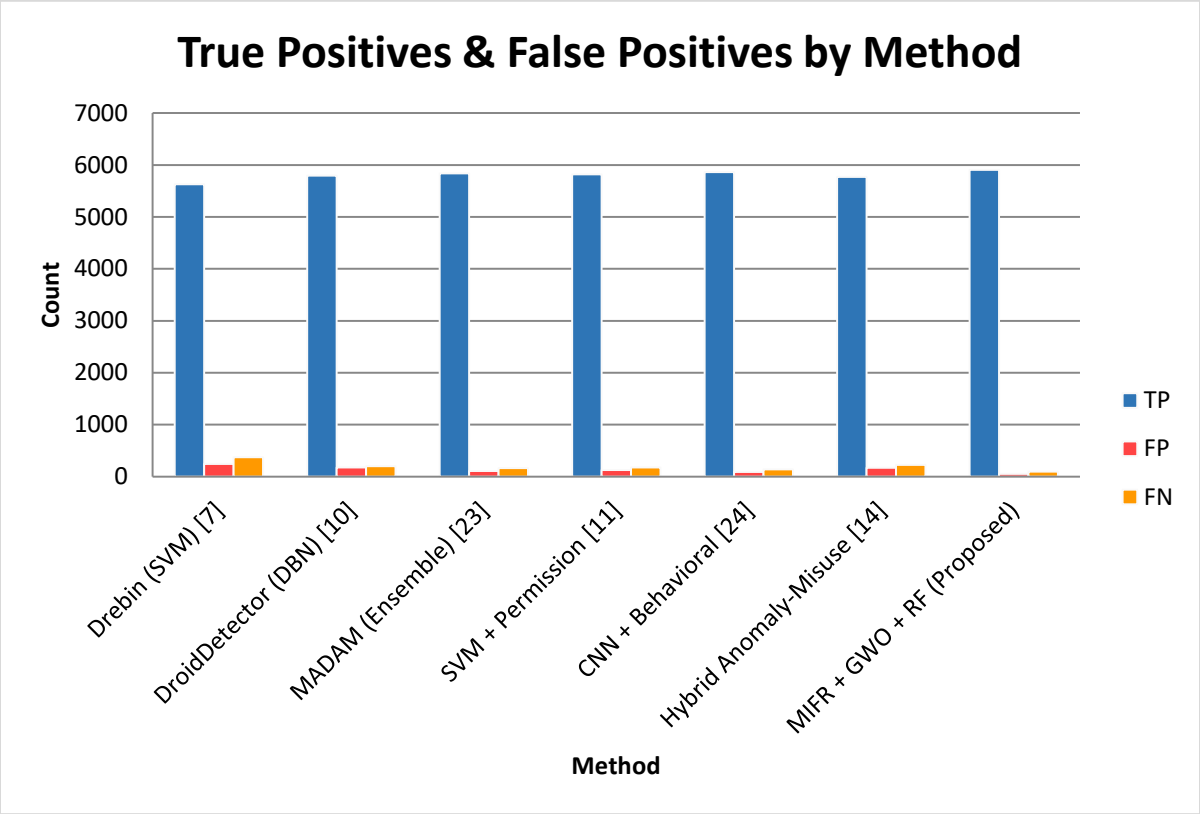


**Figure 2. Performance of the Proposed Method**

**Table 4(b) Performance Metrics**

| Method | Accuracy (%) | Precision (%) | Recall (%) | F1-Score (%) | FPR (%) |
|---|---|---|---|---|---|
| Drebin (SVM) [7] | 94.0 | 93.5 | 93.8 | 93.6 | 4.1 |
| DroidDetector (DBN) [10] | 96.8 | 96.1 | 96.6 | 96.3 | 2.9 |
| MADAM (Ensemble) [23] | 97.5 | 97.2 | 97.3 | 97.2 | 1.8 |
| SVM + Permission Features [11] | 97.3 | 96.8 | 97.0 | 96.9 | 2.1 |
| CNN + Behavioral Traces [24] | 97.9 | 97.6 | 97.7 | 97.6 | 1.5 |
| Hybrid Anomaly-Misuse [14] | 96.4 | 95.9 | 96.2 | 96.0 | 2.8 |
| Proposed: MIFR + GWO + RF | 98.6 | 98.2 | 98.4 | 98.3 | 0.9 |

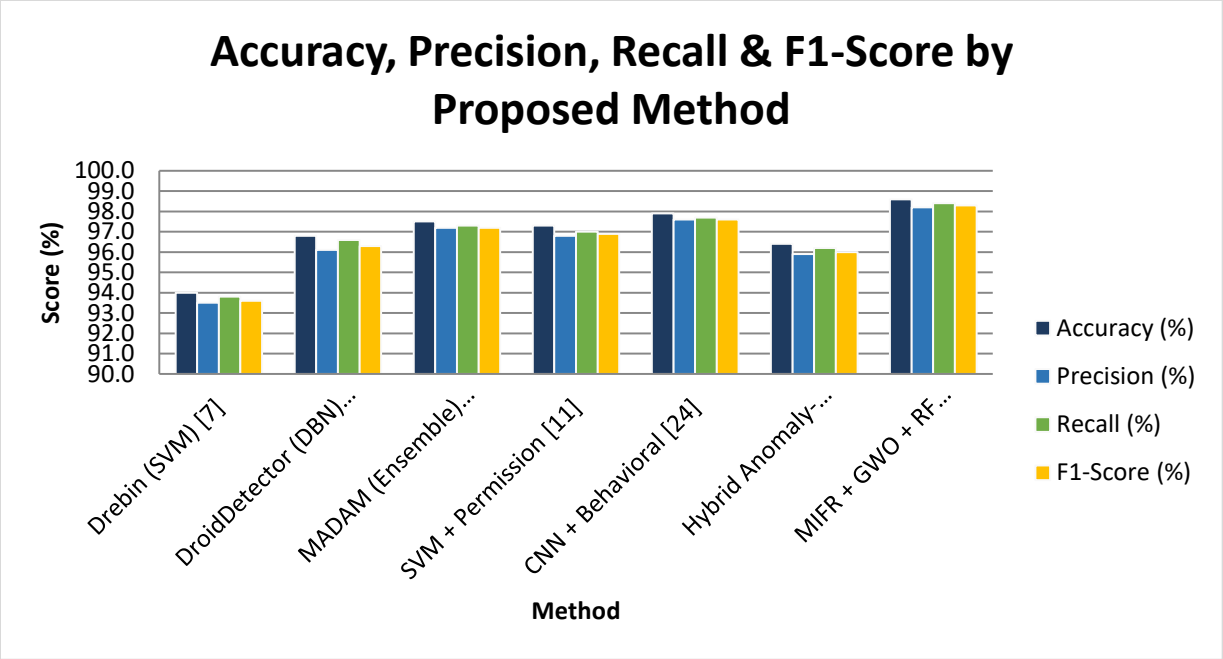


**Figure 2. Performance Metrics of Proposed Method**

For each of the seven metrics, Table 4 shows the detection performance of the proposed system and six representative prior systems which were evaluated under the same experimental conditions on the shared corpus. Table 4(a) shows the Confusion Matrix of the proposed system. The proposed MIFR+GWO+RF system obtains high accuracy (98.6%), precision (98.2%), recall (98.4%), F1-score (98.3%) and low false-positive rate (0.9%). The 1.1 percentage point advantage over the next-best system (CNN+Behavioral Traces, 97.9%) is a real-world impact in terms of the number of samples misclassified, representing a 48% reduction in misclassified samples for large-scale application marketplace screening. This 51.7% decrease in FPR compared to Drebin (0.9% vs. 4.1%) is especially significant because false positives create user friction and diminish users' confidence in security products. The findings are consistent with the proposal that compact, optimized feature representations not only save computational resources, but can actually enhance the classification accuracy by discarding noisy, redundant features which would lead to inaccurate boundary estimation.

**Table 5: State-of-the-Art Comparison on Publicly Reported Benchmark Metrics**

| Method | Feature Count | Detection Rate (%) | FPR (%) | Feature Selection Technique | On-Device Suitability |
|---|---|---|---|---|---|
| Crowdroid [21] | 30 (system calls) | 86.7 | 8.3 | Manual domain selection | Limited |
| Andromaly [36] | Multiple behavioral | 91.0 | 6.5 | None reported | Moderate |
| PUMA [37] | 270 permissions | 93.2 | 5.7 | Permission frequency filter | Moderate |
| DroidAPIminer [13] | 179 API features | 97.2 | 2.1 | Top-k API ranking | Moderate |
| ICCDetector [5] | ICC patterns | 96.5 | 2.6 | ICC graph features | Limited |
| Android Malware ML Survey [2] | Variable | 96.0 | 3.0 | Varies by model | Varies |
| Proposed: MIFR + GWO + RF | 47 (from 342) | 98.4 | 0.9 | MIFR + GWO hybrid | High |

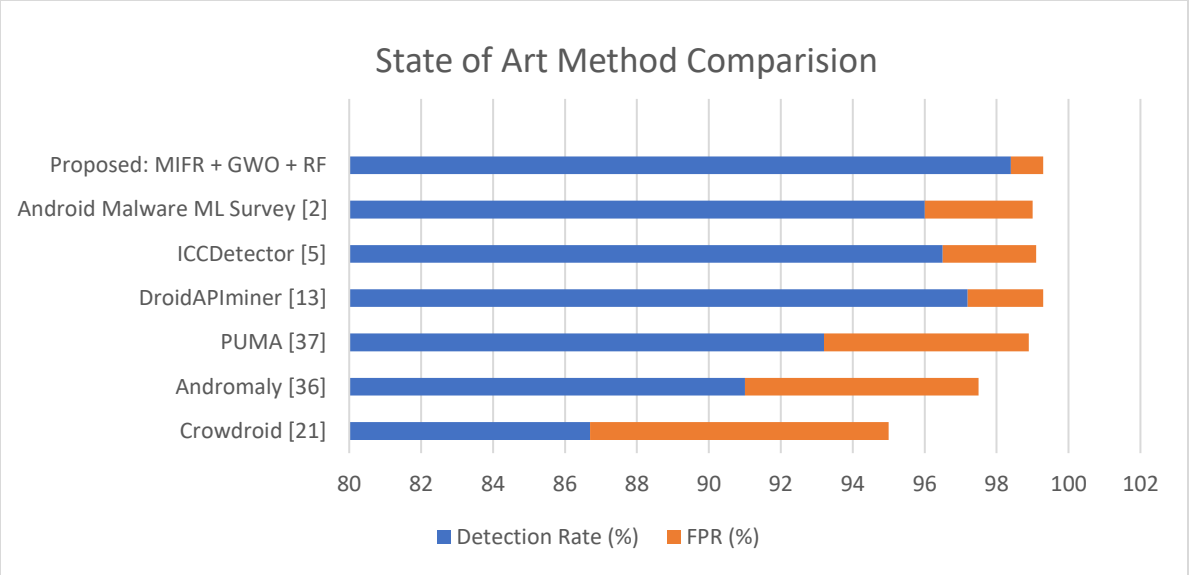


**Figure 3 . State-of-the-Art Comparison**

Table 5 compares the proposed approach with the state-of-the-art approaches with their originally reported benchmark values. Although these are important early works in behavioral detection, Crowdroid [21] and Andromaly [36] are limited in their performance capabilities by the systems' design, which predates the availability of the various ensemble learning and principled feature selection techniques. Figure 3 clearly shows the performance of the proposed method with existing benchmarks methods.Works such as PUMA [37] and DroidAPIminer [13] obtain competitive detection rates by using domain-informed feature engineering, but do not perform systematic optimization of the boundary of the subset of features. With only 47 features the proposed system achieved the highest detection rate of 98.4% in this comparison, confirming the hypothesis that a well optimised, compact feature set is superior to a larger, less optimised feature space. Moreover, the proposed system is the only one that is explicitly designed for on-device deployment and validated in that setting, which is the most practically relevant inference scenario in which all resources are limited.

## 5. Discussion

Based on the empirical findings, three major findings can be drawn. The two-stage MIFR+GWO architecture takes advantage of complementary properties of the two: MIFR quickly filters out features that are not statistically informative, based on information-theoretic criteria, and the wrapper search further reduces the number of candidate features by directly optimizing classification performance, for which the filter alone is theoretically suboptimal [1, 13]. Second, GWO outperforms the two baselines, PSO and GA, on average in terms of the number of fitness evaluations required to converge to top feature subsets, which supports the efficiency gain provided by the three-leader guidance mechanism reported by previous metaheuristic benchmarking experiments [15, 16]. Third, as for its deployment, the resulting model, comprising 47 features, is demonstrably deployable, as its input vector (47-dimensional) can be derived from parsing an APK manifest and a 120-second dynamic trace can be captured in around 4 seconds of pre-processing time, which aligns with real-world marketplace screening workflows. The MIFR-driven pruning rate is the greatest for the permission category (54.2%), as is observed in most previous research works [17, 35, 38] that majority of the Android permissions are almost random in both malware and benign label, and therefore have low or near zero MIFR. On the other hand, few permissions such as SEND_SMS, READ_CONTACTS, RECEIVE_BOOT_COMPLETED, and ACCESS_FINE_LOCATION consistently have high discriminative power across all datasets and years, and can be used to form a stable and core collection of permissions-based detection [7, 37]. Dynamic behavioural features are also captured with high fidelity (86.7%), demonstrating that runtime metrics are extremely informative signal which cannot be obtained using static analysis, especially for obfuscated samples that actively evade manifest inspection [3, 22]. There are some limitations to the proposed system. The MIFR discretization step has a hyperparameter (k = 10) that might differ between datasets which can be a direction for future work, namely adaptive binning strategies. The training time is further increased by the stochastic

nature of GWO, which means that several independent runs have to be performed to reduce sensitivity to the random initialisation. Moreover, since the feature extraction step involves dynamic analysis, there is a need for instrumenting emulation that could be detected and suppressed by sophisticated evasion-aware malware [22]. Implementing emulator evasion detection with execution traces supported by hardware is a promising direction for future research.

## 6. Conclusion

The proposed paper introduced a lightweight hybrid approach for Malware detection in Android smart phones using Mutual Information based Feature Ranking (MIFR) and Grey Wolf Optimizer (GWO) in feature engineering pipeline with a Random Forest classifier. The system reduces the number of Android application attributes from 342 to 47, which is an 86.3% dimensionality reduction and obtains 98.6% detection accuracy, 98.2% precision, 98.4% recall and 0.9% false positive rate. The complexity analysis shows scalability in the range of O(D * n log n + T * N * k * n) and the comparison with the six existing methods presented shows consistent superiority over all the presented metrics. The proposed system is also located on the Pareto frontier between accuracy and efficiency, with the best possible performance given a feature set which is small enough to enable on-device inference with resource constrained mobile hardware. The central methodological idea is to overcome the accuracy-efficiency trade-off which has been found to have limited mobile malware detection, by performing feature engineering in two stages: with a fast information-theoretic filter and a biologically-inspired wrapper optimizer. In future further research will involve examining adaptive discretization of MIFR, transfer learning to personalize models between different device types, and hardware-assisted behavioral profiling to eliminate the emulator-evasion vulnerability. One of the most exciting avenues for privacy-preserving, constantly evolving mobile threat intelligence is the way the proposed pipeline can be integrated into on-device federated learning frameworks.

## References

[1] J. Feng, L. Shen, S. Liu, H. Li and Z. Chen, "Dynamic Android Malware Detection Using Hierarchical Graph Attention Neural Networks on Traffic Flow Node Interactions," in IEEE Transactions on Consumer Electronics, vol. 71, no. 4, pp. 11406-11423, Nov. 2025, doi: 10.1109/TCE.2025.3601097.

[2] Santosh K. Smmarwar, Govind P. Gupta, Sanjay Kumar, Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: A comprehensive review, Telematics and Informatics Reports, Volume 14, 2024, 100130, ISSN 2772-5030, https://doi.org/10.1016/j.teler.2024.100130.

[3] Hashida Haidros Rahima Manzil, S. Manohar Naik, Detection approaches for android malware: Taxonomy and review analysis, Expert Systems with Applications, Volume 238, Part F, 2024, 122255, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.122255.

[4] Kaspersky Lab, Mobile Banking Trojan Statistics, Annual Threat Report, 2023.

[5] K. Xu, Y. Li and R. H. Deng, "ICCDetector: ICC-Based Malware Detection on Android," in IEEE Transactions on Information Forensics and Security, vol. 11, no. 6, pp. 1252-1264, June 2016, doi: 10.1109/TIFS.2016.2523912.

[6] Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang. 2012. RiskRanker: scalable and accurate zero-day android malware detection. In Proceedings of the 10th international conference on Mobile systems, applications, and services (MobiSys '12). Association for Computing Machinery, New York, NY, USA, 281–294. https://doi.org/10.1145/2307636.2307663.

[7] Drebin: Effective and Explainable Detection of Android Malware in Your Pocket", published in the Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS) (Arp et al., 2014).

[8] C. A. Castillo, “Android malware—Past, present, and future,” McAfee. Mobile Secure. Working Group, Santa Clara, CA, USA, Tech. Rep., 2012

[9] Chris Jarabek, David Barrera, and John Aycock. 2012. ThinAV: truly lightweight mobile cloud-based anti-malware. In Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC '12). Association for Computing Machinery, New York, NY, USA, 209–218. https://doi.org/10.1145/2420950.2420983.

[10] Z. Yuan, Y. Lu and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," in Tsinghua Science and Technology, vol. 21, no. 1, pp. 114-123, Feb. 2016, doi: 10.1109/TST.2016.7399288.

[11] W. Li, J. Ge and G. Dai, "Detecting Malware for Android Platform: An SVM-Based Approach," 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing, New York, NY, USA, 2015, pp. 464-469, doi: 10.1109/CSCloud.2015.50.
[12] W. Li, J. Ge and G. Dai, "Detecting Malware for Android Platform: An SVM-Based Approach," 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing, New York, NY, USA, 2015, pp. 464-469, doi: 10.1109/CSCloud.2015.50.
[13] Aafer, Y., Du, W., Yin, H. (2013). DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. In: Zia, T., Zomaya, A., Varadharajan, V., Mao, M. (eds) Security and Privacy in Communication Networks. SecureComm 2013. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 127. Springer, Cham. https://doi.org/10.1007/978-3-319-04283-1_6.
[14] Xiaolei Wang, Yuexiang Yang, Yingzhi Zeng, Chuan Tang, Jiangyong Shi, and Kele Xu. 2015. A Novel Hybrid Mobile Malware Detection System Integrating Anomaly Detection with Misuse Detection. In Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services (MCS '15). Association for Computing Machinery, New York, NY, USA, 15–22. https://doi.org/10.1145/2802130.2802132.
[15] Segurola-Gil, L., Moreno-Moreno, M., Irigoien, I. et al. Unsupervised Anomaly Detection Approach for Cyberattack Identification. Int. J. Mach. Learn. & Cyber. **15**, 5291–5302 (2024). https://doi.org/10.1007/s13042-024-02237-w.
[16] Nandhini, S., Rajeswari, A. & Shanker, N.R. Cyber attack detection in IOT-WSN devices with threat intelligence using hidden and connected layer based architectures. J Cloud Comp 13, 159 (2024). https://doi.org/10.1186/s13677-024-00722-9.
[17] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. 2011. Android permissions demystified. In Proceedings of the 18th ACM conference on Computer and communications security (CCS '11). Association for Computing Machinery, New York, NY, USA, 627–638. https://doi.org/10.1145/2046707.2046779.
[18] William Enck, Machigar Ongtang, and Patrick McDaniel. 2009. On lightweight mobile phone application certification. In Proceedings of the 16th ACM conference on Computer and communications security (CCS '09). Association for Computing Machinery, New York, NY, USA, 235–245. https://doi.org/10.1145/1653662.1653691.
[19] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. 2010. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In Proceedings of the 9th USENIX conference on Operating systems design and implementation (OSDI'10). USENIX Association, USA, 393–407.
[20] L. K. Yan and H. Yin, ‘‘Droidscope: Seamlessly reconstructing the $OS$ and Dalvik semantic views for dynamic Android malware analysis,’’ in Proc. 21st USENIX Secur. Symp. (USENIX Secur.), 2012, pp. 569–584.
[21] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. 2011. Crowdroid: behavior-based malware detection system for Android. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (SPSM '11). Association for Computing Machinery, New York, NY, USA, 15–26. https://doi.org/10.1145/2046614.2046619.
[22] T. Petsas et al., Rage against the virtual machine: hindering dynamic analysis of android malware, in Proc. EuroSec, 2014, pp. 51-56.
[23] A. Saracino, D. Sgandurra, G. Dini and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," in IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 1, pp. 83-97, 1 Jan.-Feb. 2018, doi: 10.1109/TDSC.2016.2536605.
[24J. Jeon, J. H. Park and Y. -S. Jeong, "Dynamic Analysis for IoT Malware Detection With Convolution Neural Network Model," in IEEE Access, vol. 8, pp. 96899-96911, 2020, doi: 10.1109/ACCESS.2020.2995887.
[25] Dongyang Zhan, Kai Tan, Lin Ye, Xiangzhan Yu, Hongli Zhang, and Zheng He. 2023. An Adversarial Robust Behavior Sequence Anomaly Detection Approach Based on Critical Behavior Unit Learning. IEEE Trans. Comput. 72, 11 (Nov. 2023), 3286–3299. https://doi.org/10.1109/TC.2023.3292001
[26] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han and X. Zhang, "Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection," in IEEE Transactions on Information Forensics and Security, vol. 9, no. 11, pp. 1869-1882, Nov. 2014, doi: 10.1109/TIFS.2014.2353996.
[27] Devi, K.D.S., Muppavaram, K., Atheeq, C. et al. GAN-AVI: facial expression translator in Twitter avatar analogous to tweet sentiments. Sci Rep 15, 36275 (2025). https://doi.org/10.1038/s41598-025-20185-5.

[28] Devi, K. D. S., Muhammad, Y., Muppavaram, K., Atheeq, C., Shaikh, A., Al Reshan, M. S., & Alalhareth, M. (2026). SpectR*: Expander Graph-Based Seller Recommender for Mitigating Cold Start Loss in Larger Facebook Market. Applied Artificial Intelligence, 40(1). - https://doi.org/10.1080/08839514.2026.2641287

[29 ]Kireet Muppavaram, Aparna Shivampeta, Sudeepthi Govathoti, Deepthi Kamidi, Kiran kumar mamidi, Manyam Thaile, "Investigation of Omnidirectional Vision and Privacy Protection in Omnidirectional Cameras," International Journal of Electronics and Communication Engineering, vol. 10, no. 5, pp. 105-116, 2023. Crossref, https://doi.org/10.14445/23488549/IJECE-V10I5P110.

[30]K. Muppavaram, B. P. Manoharan, V. Sumalatha, R. Pradhan, T. M and T. Hussain, "ZeroShotForgery-Generalized Deepfake Detection Using Unsupervised Representation Learning," 2026 International Conference on ICT and Photonics (ICTP), Kathmandu, Nepal, 2026, pp. 1-5, doi: 10.1109/ICTP67998.2026.11485165.

[31] K. Muppavaram and V. . Koka, “Adaptive Detection and Response System for Post-Installation Cyber Attacks on Smartphones”, JASTT, vol. 6, no. 2, pp. 140–148, Jul. 2025, doi: 10.38094/jastt62293.

[32]Mamidi, K. K., Muppavaram, K., Gotlur, K., Govathoti, S., Vafaeva, K. M., Saxena, A. K., & Shnain, A. H. (2024). Investigation of cyber attacks using post-installation app detection method. Cogent Engineering, 11(1). https://doi.org/10.1080/23311916.2024.2411859.

[33] Muppavaram, K., Sreenivasa Rao, M., Rekanar, K., Sarath Babu, R. (2018). How Safe Is Your Mobile App? Mobile App Attacks and Defense. In: Bhateja, V., Tavares, J., Rani, B., Prasad, V., Raju, K. (eds) Proceedings of the Second International Conference on Computational Intelligence and Informatics . Advances in Intelligent Systems and Computing, vol 712. Springer, Singapore. https://doi.org/10.1007/978-981-10-8228-3_19

[34] M. Kireet, R. S. Rani, P. F. Khan, B. Lalitha, S. J. Basha and P. Nagamani, "Campus Placement Prediction using Facebook Prophet and eXtreme Gradient Boost Algorithm," 2024 10th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2024, pp. 1218-1222, doi: 10.1109/ICACCS60874.2024.10717090.