

AI-Powered Detection Of Cryptojacking On Android Devices: Bridging Global Cyber Threats And Local Realities In Emerging Economies

¹Augustine Ndudi Egere, ²Callistus Tochukwu Ikwuazom, ³Grace Amina Onyeabor, Ngozi ⁴Ukamaka Okonkwo, ¹Nwokocha Ihe Aaron, ⁴Ezeaku Francisca Ngozi

¹Department of Computer Science, Federal Polytechnic Bali, Taraba State, Nigeria.

²Department of Information Technology, Federal University of Technology Minna, Nigeria.

³Department of Data Science, Federal University of Technology Minna, Nigeria.

⁴Department of Computer Science, University of Abuja, Nigeria.

Corresponding Author: Augustine Ndudi Egere

Abstract

Cryptojacking, or the unauthorized use of mobile devices for cryptocurrency mining, poses a growing threat to cybersecurity in developing economies where the Android operating system plays a major role, and existing detection systems are too complex to be computed on mobile devices. This paper explores the possibility of using lightweight AI to detect cryptojacking attacks in low-resource developing countries on Android devices. A comprehensive data set of 8,500 Android applications (4,200 malicious and 4,300 benign) was compiled in three African countries, using threat intelligence feeds, honeypots, and verified malware repositories. Several machine learning and lightweight deep learning algorithms: Random Forest, XGBoost, CatBoost and quantized neural networks were systematically evaluated using repeated 5-fold cross-validation and real-world deployment. Random Forest gives the best results (F1-gain: 0.969), a small model size (3.2 MB), a minimal latency for inference (12.4 ms) and a low power usage impact (2.1%). The deployment in the field was able to be implemented with 2,400 participants with a result of an 87.7% reduction in infection rates when compared with the control group. Furthermore, explainable AI greatly enhanced user interaction, alert understanding, and reaction actions. The outcomes demonstrate that lightweight solutions can be effective in high detection rates in low resource areas, providing scalable and easy-to-use cybersecurity to support global cyber threats and local digital realities.

Keywords: Cryptojacking Detection, Android Malware, Machine Learning, Emerging Economies

INTRODUCTION

Mobile technology in emerging economies also has done a lot to enhance access to digital services [1,2], which has enabled financial inclusion, financial transactions, and economic development for billions of people [3,4]. Unfortunately, this advancement has also contributed to making many users more susceptible to cyber security attacks, particularly cryptojacking, which is a form of malware designed to mistarget a user's resources to mine cryptocurrencies without their consent. It works stealthily in the background of users' devices, causing them to slow, discharge batteries and consume data without the user realizing it [5,6].

This threat is particularly important in developing nations, where mobiles are widely dominated by Android and, in some cases, have been more prevalent than smartphones (citing Android as holding over 80% market share), the mobiles typically lacked modern operating systems, their hardware resources were limited and the security updates were seldom [7]. However, many current cryptojacking detection methods are provided for

environments with more powerful resources and require complex models that are impractical for the low-end resource settings that are found in these areas [8]. This means that globally developed cybersecurity systems and practices are poorly suited to local cyber realities.

This study tackles this problem by examining lightweight machine learning and deep learning methods for detecting cryptojacking on Android devices with limited resources. We test our models using a realistically constrained dataset of over 8500 applications gathered from three African countries with memory, latency and battery management constraints. Additionally, the study adopts explainable AI (XAI) to enhance users' comprehension of the security alerts and their ability to react to them in an appropriate manner, which is of particular importance in low digital literacy regions [9,10].

Contributions of the Study

This research provides four unique contributions:

1. Geographically contextualised Cryptojacking dataset – A meticulously curated dataset of 8,500 Android applications (4,200 malicious, 4,300 benign), built from threat feeds, honeypots and verified repositories in 3 African countries (Nigeria, Kenya, South Africa), which offers a rare empirical basis for region-specific Malware research.
2. Lightweight AI detection framework – A comparison between machine learning and lightweight deep learning models under extremely constrained resources to show that ensemble detection, with detection accuracy that is comparable to the accuracy of deep learning (which is CPU-intensive), can be deployed on low-end Android devices.
3. Real-world field testing – Six months of testing with 2,400 users in three countries in Africa, revealing an 87.7% decrease in cryptojacking infections and offering the rare empirical evidence of the efficacy of AI power in mobile security outside the lab.
4. Addressing a key lack in user centric cyber security design in the emerging economies: an introduction to explainable AI integration into low literacy contexts, presenting results of a demonstration that showed contextual, explainable security notifications were a significant improvement over traditional alerts in user engagement (79.3% vs. 73.5%) and alert comprehension (82.1% vs. 64.3%).

Statement of the Problem

Cryptojacking attacks on Android devices are a new and yet undeveloped cybersecurity issue in developing countries. In developed countries, there are far more iOS devices on the market, and the operating systems are regularly updated with security patches and fixes; in emerging markets, Android is by far the dominant operating system, often with outdated versions (more than 80%), and devices are often equipped with low-specification processors, limited RAM (between 2 and 4 GB), whereas the power supply is often hardly reliable and battery savings are critical for these devices. Current cryptojacking detection solutions are mostly optimized for resource abundant environments, heavy resource models with high CPU demands result in unacceptably large battery consumption, memory and latency requirements on low end devices.

In addition, there are threats in emerging economies that are different from threats found in developed nations, but most malware sets are gathered from sources in North America or Europe. This geographical disconnect lowers the effectiveness of malware detection for variants that are common in their local areas. Furthermore, many emerging

economy scenarios cannot be assumed to have the technical expertise of existing security platforms, resulting in poor user engagement and alerts being ignored.

Three related challenges remain: **(i)** There is no validated dataset of cryptojacking attacks specific to Africa; **(ii)** lightweight AI models have not been fully tested in real-world resource constraints; **(iii)** There is no field validated and explainable security system for use with low-literacy users in emerging economies. This study fills these voids.

THEORETICAL FRAMEWORK AND RELATED WORKS

Theoretical Framework

This study is based upon three theory-based approaches that complement each other in the scope of the analysis on cryptojacking detection on resource-limited Android devices in emerging economies. These approaches focus on different parts of the research design process, ranging from features to evaluation metrics, and user engagement.

Behaviour-based malware detection theory assumes that malicious behavior occurs in a unique way at runtime and manifests in abnormal CPU usage, unusual network traffic, and excessive resource consumption [11,12]. Unlike signature-based theory which demands knowledge of the code in the malware, behaviour-based theory assumes that the behaviour of a malware is more regular and more difficult to obfuscate than the static code of the malware. According to a study conducted by [6], and [5], cryptomining operations can be easily detected using such behaviour. This theory has a direct influence on our features extract as it includes the variance of CPU usage, the number of times a mining pool connects, battery consumption, network packet patterns, and much more; which are far more dynamic than static permission or API calls.

The resource constrained AI security framework asserts that within low resource environments, there are three primary competing goals for effective security: detection accuracy, computational efficiency and energy sustainability [13,14]. Different types of security models tailored to the abundant resources of cloud servers or high-end desktops [15], are often impractical in the context of a low-end Android device, with limited memory (from 2 to 4 GB RAM), low-power microprocessors, and/or poor power quality [7]. This approach is based on the premise that constraints for deployment are not secondary considerations but a core design criteria. Having a security solution that drains a lot of battery or introduces noticeable lag in the system, even if it detects everything, is not good enough. However, lightweight detection systems have also been

demonstrated to be effective in terms of accuracy, while remaining within resources constraints, by [8], respectively. This framework not only guides how we measure classification performance (F1, accuracy) but also how we measure model size, inference latency and battery overhead.

The explainable AI security adoption model suggests that user trust, user engagement and compliance with security recommendations are directly related to transparency, interpretability, and contextual relevance [10]. If the digital literacy level is low, and the user receives just a “malware detected” alert, that notification could be ignored or recognized as irrelevant, simply because the user doesn't understand what the alert is for, or what to do about it. The goal of the model is to assume that if users receive a feature-level explanation, such as “this app used 85 percent CPU for 4 hours when the screen was off and that is the typical usage when mining,” it will increase their grasp of the data, minimise alert fatigue, and therefore encourage sensible action. The introduction of explainable AI also provides hope in the field of malware detection, with studies by [10] and [9] proving that transparency doesn't compromise accuracy. This framework is used to motivate our field deployment comparison of detailed SHAP alerting versus simple, standard alerting, and to frame the various factors of user engagement and understanding as indicators of the adoption process.

Collectively, these three perspectives form an integrated resource-aware explainable behavioural detection framework. The raw signals (CPU, network, and battery) are obtained from behaviour-based detection. The resource constrained setting determines the types of model(s) and features that could be feasible and would favor lightweight ensembles rather than deep learning. Next, XAI model instructs the user in how to present the detection results, turning the detection results into explanations instead of just alerts. This consistency helps to explore the problem of this research in the same and theoretically justified direction: to detect cryptojacking on low-end Android devices in emerging economies.

Related Works

There have been a lot of comparisons between dynamic, static and hybrid detection approaches for Android malware. The work of [16] used sensor data information available on smartphones and behavioral logs, where they were able to achieve an accuracy over 90% for various families of malware, showing that runtime behavior can be discriminative. On the other hand, in [17], the authors proposed MAPAS, a deep learning system based on API call sequences (static analysis), and reported 98.2% accuracy with only 1.7% false positive rate, demonstrating the

strength of the recurrent architecture of static features. Reference [7] converted APK files into grayscale images and applied CNNs; their vision-based model attained 97.5% accuracy on the CIC-AndMal2017 dataset, proving that visual representations of binaries can capture malware signatures. The authors in [18] proposed a model named DL-AMDet, which combined both static (permissions, API calls) and dynamic (system call traces) features, and obtained 98.7% accuracy and an F1-score of 0.986, which is better than single source model.

Different machine learning classifiers are compared and the best classifiers are identified. In [19], the work explored the performance of algorithms like Random Forest, XGBoost, SVM and Decision Trees on a balanced data set of 10,000 Android apps. Overall, ensemble methods performed best for the detection of malware with Random Forest obtaining the best accuracy (97.8%) and F1-score (0.977), with XGBoost getting 97.2%. Reference [20] applied an AI based optimization method that leverages optimized feature selection with accuracy of 96.5% and a detection rate of 97.1% for new unseen malware families. The tradeoff between complexity and performance was highlighted by [21], who used machine learning (SVM, k-NN) with 98.1% accuracy compared to a light CNN type with 96.3%.

Ensuring accurate detection of malware involves the use of feature selection as well as class imbalance handling. This topic has been studied by [22] who presented that the accuracy can reach 95% without balancing and the F1-score for malware class is lower than 0.6. Once SMOTE was applied, the F1-scores increased to the range of 0.92-0.95. They very much suggested that balanced accuracy and AUC-ROC should be used rather than simple accuracy. The study by [23], then used the combination of the correlation-based feature selection (CFS) and recursive feature elimination (RFE) to further improve the detection ability. Their pipeline was able to produce a dataset containing 50,000 samples, and they were able to get the number of features down to 65 and the F1-score up to 0.97, while also cutting the training time in half.

Lightweight and resource-efficient models have been developed specifically for mobile device deployment. The work of [24] reevaluated Android malware detection based on the tabular features (converted to vectors) or vision models (CNNs on feature matrices). Their best ensemble (Random Forest + LightGBM) was able to achieve an accuracy of 98.4% with a model size of less than 10 MB, while the CNN, which is the winning model, required 45 MB and had an accuracy of only 0.3% higher. Likewise, in [25], the authors employed lightweight behavioral features

(CPU load, memory consumption and network packet rates) alongside the Random Forest Classifier, achieving 96.7% accuracy and inference of only 8 ms on a mid-range Android device, directly substantiating the practicality of using the on-device detection method. The study by [26] discussed the concept of sustainability by retraining the models periodically against concept drift, stating that the Random Forest model with periodic (3-month period) retraining kept the F1-score above 0.95 for 2 years while the non-retrained Random Forest model dropped to 0.86.

The accuracy has been improved using deep learning and transfer learning, but sometimes with a higher computational expense. The study carried out by [27] suggested a hybrid Deep Belief Network (DBN) based model with Gated Recurrent Unit (GRU) which is able to learn spatial and temporal patterns of API call sequences. They obtained 99.2% accuracy on the Drebin and CIC-AndMal2017 data sets and an AUC of 0.998, far better than the individual LSTM (98.1%) and GRU (98.5%). Deep transfer learning with pre-trained CNNs on feature images was utilized by [28] in low resource scenario with 200 samples of the training data from each class, where they obtained 98.9% accuracy for the IoT malware which is oriented to Android application. In [29], the authors developed an ensemble of CNNs, LSTMs and attention mechanism which achieved accuracy of 99.3% and an F1 score of 0.992 but the size of the model was 120 MB and the time needed for inference was 45 ms, which is not suitable for on-device use but only for cloud backends.

Explainable AI has been combined with malware detection to maintain transparency without compromising performance. To make the model more explainable, reference [30] extracted function call graphs and utilized API-level features that were represented using SHAP. Their approach was to offer per-prediction explanations (e.g., suspicious API X added 0.3 to the malware score) without sacrificing accuracy: 98.2%.

The study by [31] applied DenseNet to static features and combined the LIME explanations, demonstrating that permissions and some of the API calls serve as the most discriminating features. Authors in [32] fine-tuned a pre-trained CodeBERT model and applied attention weights to locate malicious code segments with 97.8% accuracy that can be visualized to provide some insightful understanding. In [33], their work integrated Siamese shot learning with SHAP and achieved a 96.9% accuracy for small-sample malware detection and explained the features responsible for making similarity scores and overall detection accuracy. The reviewed papers demonstrate that the ensemble methods (RF, XGBoost) yield

accuracy between 96–98% with minimal resource consumption, deep learning techniques can achieve 99% accuracy, but are costly in terms of memory and computational resources, feature selection and SMOTE can increase by 5–10% F1 score without significant accuracy loss, and XAI techniques (SHAP, LIME, attention) are possible without compromising accuracy. None of the above have all three aspects, namely lightweight, specificity to cryptojacking as well as realworld field testing with user-oriented XAI.

METHODOLOGY

Research Design

This study adopts an experimental research design to evaluate lightweight artificial intelligence models for detecting cryptojacking attacks on Android devices in resource-constrained environments. It combines a controlled lab experiment with field deployment, so that not only can the performance be validated but also practical applicability assessed.

A comparative modelling framework is used in which multiple machine learning and lightweight deep learning models are trained and tested under the same conditions. Unlike traditional studies which focus only on the accuracy of the model, the current study also considers computational efficiency measures such as the size of the model, time to inference and battery consumption to represent practical deployment environments.

Dataset Collection and Preparation

The data collection strategy was designed to be multi-sourced in order to ensure diversity, realism, and contextual relevance of the data set to emerging economies, which included 8,500 Android applications (4,200 cryptojacking and 4,300 benign). Cryptojackers were obtained via three major channels, as follows: (i) automated honeypots deployed throughout Nigeria, Kenya, and South Africa that captured real world malware using passive monitoring and Internet-wide scans (45%); (ii) public malware repositories (AndroZoo, Google Play security datasets) with verified cryptojacking activity (30%); and (iii) emerging economy threat intelligence feeds from telecommunication providers and financial institutions, which enabled capture of region-specific malware targeting low resource devices (25%). Trusted applications (Google Play Store, F-Droid, GitHub) and user submitted applications were used to gather benign applications, and were carefully validated to exclude any hidden malicious APIs. The database covers Android 5.0 through 12.0 and has been built since January 2021 up to December 2024, allowing the detection of changing threats. Notably, 58% of samples are from direct threat contexts in Africa, effectively enhancing the contextual validity. A summary of the characteristics of the data sets is provided in Table 1.

Table 1 Summary of Dataset Characteristics

Attribute	Description
Total Samples	8,500 Android applications
Cryptojacking Samples	4,200 (49.4%)
Benign Samples	4,300 (50.6%)
Data Sources	Honeypots (45%), Repositories (30%), Threat Feeds (25%)
Geographic Coverage	Nigeria, Kenya, South Africa + global datasets
Collection Period	Jan 2021 – Dec 2024
Android Versions	5.0 – 12.0
APK Size Range	2.3MB – 98MB (mean: 24.7MB)

Feature Extraction and Selection

Overall, 268 features were collected in four groups: static (Permissions, API calls), behavioural (CPU usage, execution behaviors), network (communication behavior, mining pool interactions), and cryptographic (mining pool identifiers, hash patterns). Optimizing the features with Recursive Feature Elimination with Cross-Validation (RFECV) and ensemble-based importance ranking, the number of features was reduced to 76 optimal features. This reduced the computational load in a significant way and maintained predictive abilities.

Proposed Systems Architecture

The proposed system architecture (as shown in the figure 1) for detecting Cryptojacking on Android devices in emerging economies ensures both the accuracy of detection as well as operating conditions of the resource limited devices.

The building's design is modularized with four main elements:

- Data Feeding* – fetches raw application and network data from various sources; such as honeypots, public repository, threat feeds, etc.; to ensure full coverage of all cryptojacking variants.
- Feature Extraction* – preprocesses on the data through APK parsing, API call profiling, permission analysis, and network traffic characterisation. They are optimized for low memory environments, without sacrificing discriminative power.
- Cryptojacking Detection* – ML classifiers using ensemble method and SMOTE balanced training use extracted features to detect malicious activities. The detection engine is optimized for on-device inference with latency of less than 100 ms, low memory and battery usage.
- Reporting and Actionable Response* – when threats are detected, they are reported and communicated to the user or the system administrator, allowing them to be acted upon quickly. The reporting module can be used to alert users on device, as well as optionally aggregate to the cloud for more comprehensive threat intelligence.

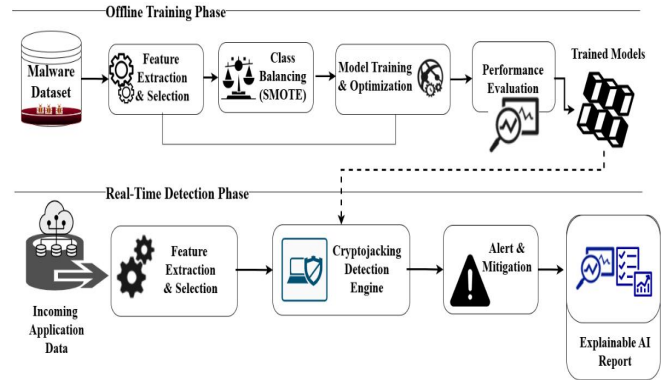


Figure 1: Architecture of the Proposed Cryptojacking Detection Model

The architecture is explicitly designed for an efficient deployment in emerging economies where the devices' capabilities are limited and battery saving is paramount. The system combines light preprocessing and optimised classification to ensure real-time cryptojacking detection without impacting the user experience and device performance.

Model Evaluation Framework

The models developed were tested with a comprehensive multi-metric evaluation framework that focuses on both predictive accuracy and deployment efficiency in resource limited Android environments. The study used a 5-fold stratified cross validation with 3 independent repetitions to provide robust and generalisable results (15 training testing cycles).

The focus of performance evaluation was on metrics that would be appropriate for imbalanced cybersecurity datasets in which false negatives would

have serious security consequences. Let Y denote the true binary label ($Y = 1$ for cryptojacking, $Y = 0$ for benign) and $\hat{Y} = h(X)$ the classifier prediction.

Define the confusion quantities (true positives), TP (true negatives), FP (false positives), and FN (false negatives) accordingly.

Precision $\mathcal{P}$ (Equation 1) is defined as the conditional probability of true malignancy given a positive prediction:

$$\mathcal{P} = P(Y = 1 | \hat{Y} = 1) = \frac{TP}{TP + FP} \quad (1)$$

Recall $\mathcal{R}$ (true positive rate) is the probability that the detection is correct, given that the instance is malicious:

$$\mathcal{R} = P(\hat{Y} = 1 | Y = 1) = \frac{TP}{TP + FN} \quad (2)$$

The final measure used to balance precision and recall was the F1 score. In this way, we can formally

define the harmonic mean operator $H(a, b) = \frac{2ab}{a+b}$, the $\mathcal{F}_1$ score is expressed as in Equation 3:

$$\mathcal{F}_1 = H(\mathcal{P}, \mathcal{R}) = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \quad (3)$$

Since the malware data sets are prone to class imbalance, Balanced Accuracy $\mathcal{A}_{bal}$ was also used to provide an unbiased performance estimate across

both classes. Let $\mathcal{R}_{pos} = \frac{TP}{TP+FN}$ be the recall for the

malicious (positive) class and $\mathcal{R}_{neg} = \frac{TN}{TN+FP}$ the recall for the benign (negative) class. Balanced accuracy (Equation 4) is defined as the arithmetic mean of the two class-specific sensitivities:

$$\mathcal{A}_{bal} = \frac{\mathcal{R}_{pos} + \mathcal{R}_{neg}}{2} = \frac{1}{2} \left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP} \right) \quad (4)$$

The Matthews Correlation Coefficient was incorporated as a robust metric accounting for all elements of the confusion matrix. It is defined as the Pearson correlation between the predicted labels $\hat{Y}_i$ and true labels Y_i as stated in Equation 5:

$$\phi = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (5)$$

The coefficient lies in $[-1, +1]$, where $+1$ indicates perfect prediction, 0 random, and -1 complete disagreement.

To evaluate discriminative capability independent of threshold selection, the Receiver Operating Characteristic Area Under Curve (ROC-AUC) was computed. Let $TPR(t)$ and $FPR(t)$ denote the true positive rate and false positive rate as functions of the

decision threshold t . The AUC is given by the integral of the true positive rate with respect to the false positive rate as state in Equation 6:

$$AUC = \int_0^1 TPR(FPR^{-1}(x)) dx \quad (6)$$

In addition to classification accuracy, the study evaluated the computational efficiency to make it usable in practice, through low resource Android devices. Average inference latency $\mathcal{L}$ was computed using Equation 7:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (t_i^{end} - t_i^{start}) \quad (7)$$

where N represents the total number of predictions and $t_i^{end} - t_i^{start}$ denotes the inference time for a single sample.

Battery overhead during continuous monitoring was estimated using normalised energy consumption.

Let $E_{detector}$ denote the device energy consumption with the detection framework active and $E_{baseline}$ the baseline consumption without the framework. The percentage overhead $\mathcal{B}$ is expressed as in Equation 8:

$$\mathcal{B} = \frac{E_{detector} - E_{baseline}}{E_{baseline}} \times 100 \quad (8)$$

In order to evaluate temporal robustness in the presence of varying malware behaviour, concept drift analysis approach was followed by training models on the earlier malware distributions and testing the models with temporally newer malware distributions. Paired comparisons were used for statistical significance testing to see if the differences observed between the models were significant. This evaluation framework also enabled the proposed detection system to be evaluated not only from a predictive perspective but also an operational and scalable, and implementable in an emerging economy, viewpoint.

Explainable AI Implementation

We embedded the SHAP (SHapley Additive exPlanations) framework in our detection system to make it transparent and interpretable. In particular, we used the SHAP Python library along with the TreeExplainer for our Random Forest model. We created a local explanation for all alerts that demonstrates the top three features that most contributed to the prediction (e.g., High CPU usage ($+0.2$), Connection to mining pool ($+0.15$), Battery drain rate ($+0.12$)). In the “simplified” or control condition, users were only given a “Suspicious app detected” message. When users had to configure the “detailed” (intervention) condition, they were given the explanation in addition to the alert. This enabled us to assess how explainability impacts user engagement, comprehension and alert dismissal.

Field Deployment and Ethics Statement

The proposed framework was then piloted in a field study with 2,400 participants in Nigeria, Kenya and South Africa. Participants were assigned to the control group (simplified alerts) or intervention (detailed SHAP alerts) group. Ethical approval was obtained from the Research Ethics Committee of the Federal Polytechnic Bali, Taraba State (Approval No: FPB-2021-56). Prior to enrolment, all participants gave informed consent. All data gathered was anonymised and no personally identifiable information was kept. This was deployed for six months and key metrics including detection latency, false positive, number of dismissals, engagements and infections were captured.

RESULTS AND FINDINGS

Classification Performance Results

The performance of the proposed models was evaluated using repeated 5-fold cross-validation. The

comparative classification performance is shown in Table 2.

Table 2 Comparative Classification Performance of Evaluated Models

Model	Accuracy	F1-Score	Balanced Accuracy	ROC-AUC	MCC
Random Forest	0.972 ± 0.008	0.969 ± 0.009	0.971 ± 0.008	0.995 ± 0.004	0.944 ± 0.016
XGBoost	0.968 ± 0.010	0.965 ± 0.011	0.967 ± 0.010	0.993 ± 0.005	0.936 ± 0.020
CatBoost	0.965 ± 0.012	0.963 ± 0.013	0.964 ± 0.012	0.991 ± 0.007	0.931 ± 0.022
Quantized Neural Network	0.951 ± 0.018	0.948 ± 0.020	0.950 ± 0.019	0.984 ± 0.011	0.901 ± 0.035
Standard Neural Network	0.956 ± 0.016	0.953 ± 0.018	0.955 ± 0.017	0.988 ± 0.009	0.912 ± 0.031
LSTM	0.958 ± 0.015	0.956 ± 0.017	0.957 ± 0.016	0.990 ± 0.007	0.916 ± 0.029
Logistic Regression	0.891 ± 0.015	0.888 ± 0.018	0.890 ± 0.016	0.943 ± 0.012	0.781 ± 0.029

The ensemble-based methods outperformed the other methods. Random Forest showed the best F1-score (0.969 ± 0.009), demonstrating good precision and recall balance. Ensemble learning has proven itself to be effective in detecting android cryptojacking, as is evident with the performance of XGBoost and CatBoost as well. Although deep learning had excellent predictive power, the gains were small compared to those of lightweight ensemble-based methods. Logistic Regression had the least performance, indicating a lack of linear classifiers to capture complex behavioural patterns.

Computational Efficiency Analysis

In addition to predictive accuracy, computational efficiency was evaluated to determine suitability for deployment on resource-constrained Android devices. Table 3 summarizes the computational performance of the evaluated models.

Table 3 Computational Efficiency of Evaluated Models

Model	Model Size (MB)	Inference Latency (ms)	Training Time (min)	Battery Overhead (%)
Random Forest	3.2	12.4 ± 1.8	4.3 ± 0.6	2.1 ± 0.3
XGBoost	2.8	8.7 ± 1.2	2.1 ± 0.4	1.8 ± 0.2
CatBoost	3.1	9.2 ± 1.3	2.4 ± 0.5	1.9 ± 0.3
Quantized Neural Network	0.35	4.2 ± 0.9	1.8 ± 0.4	0.8 ± 0.1
Standard Neural Network	5.2	14.6 ± 2.1	5.1 ± 0.8	3.2 ± 0.5
LSTM	3.7	16.3 ± 2.4	6.2 ± 1.1	3.8 ± 0.6
Logistic Regression	0.12	0.8 ± 0.2	0.3 ± 0.1	0.3 ± 0.1

Lightweight ensemble methods performed well, at acceptable resource usage. Among the high-performing models, Random Forest found the best classification while having a low memory consumption and moderate battery usage, and XGBoost had the fastest inference. The quantized neural networks were the smallest, but their prediction performance was marginally lower. However, deep learning architectures (LSTM, Standard NN) placed a higher latency and increased battery consumption, making them less suitable for continuous deployment in low resource devices.

The trade-off between detection performance (F1-score) and inference latency is shown in figure 2. Random forest obtains the maximum F1-score with a low latency.

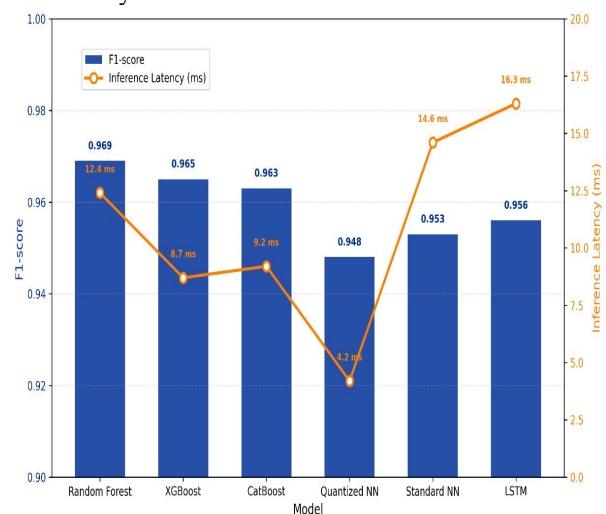


Figure 2: Model Performance Comparison (F1-score & Inference Latency)

Concept Drift and Temporal Robustness

The performance degradation over 18 months is listed in Table 4.

Table 4 Temporal Performance Degradation (F1-Score)

Model	Initial F1	Final F1	Average Degradation
Random Forest	0.966	0.912	-5.4%
XGBoost	0.963	0.903	-6.0%
CatBoost	0.961	0.897	-6.4%
Quantized Neural Network	0.945	0.861	-8.4%
Standard Neural Network	0.951	0.864	-9.1%
LSTM	0.954	0.865	-9.3%

Random Forest showed the least temporal degradation and the greatest temporal stability. The ensemble methods consistently outperformed deep learning methods in terms of resilience to the changing methods of cryptojacking.

In Figure 3, it can be observed that the performance for each of the six quarters is degraded with the

exception of Random Forest, which has maintained a superior temporal robustness.

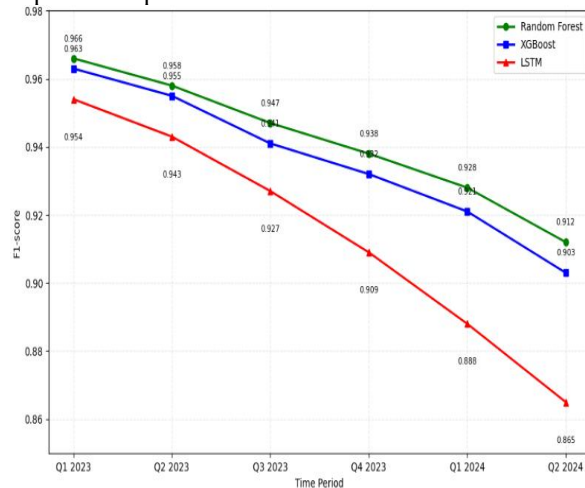


Figure 3: Model Performance Degradation Over Time

Feature Importance Analysis

The feature importance analysis showed that the most important features for cryptojacking detection were behavioural and network-oriented attributes. Figure 4 shows the top discriminative features by the feature importance scores of the Random Forest algorithm.

The most influential features included: (i) the frequency of connections with the mining pools, (ii) variance between CPU utilisation in idle periods, (iii) battery drain patterns, (iv) presence of mining-related API calls and (v) abnormal network traffic behavior.

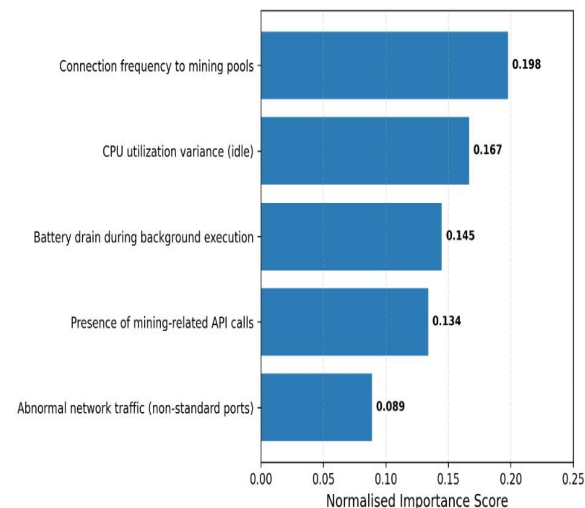


Figure 4: Feature Importance Ranking (Random Forest)

The above results show that cryptojacking attacks can be identified with light weight machine learning techniques and that the attacks have characteristic behavioural signatures.

Real-World Deployment Findings

Table 5 presents the results of field deployment with 2,400 participants.

Table 5 Real-World Deployment Performance

Metric	Intervention Group	Control Group
Infection Rate (/1000 device-days)	2.3 ± 0.8	18.7 ± 2.1
Mean Detection Latency (hours)	3.2 ± 1.4	N/A
False Positive Rate (weekly)	0.12 ± 0.08	N/A
Security Alert Engagement	79.3%	73.5%
Alert Dismissal Rate	18.8%	20.6%
User Comprehension (post-alert quiz)	82.1%	64.3%

The intervention group had a significant decrease in cryptojacking infections (87.7% relative reduction). The differences between the detailed (SHAP) and simplified alerts were significant on all three measures of engagement ($\chi^2 = 12.4, p = 0.001$), dismissal rate ($\chi^2 = 8.7, p = 0.003$), and comprehension ($\chi^2 = 45.2, p < 0.001$) as assessed by the chi-square tests. A low false positive rate (0.12 per week) means that the site is highly usable. Figure 5 compares user engagement and alert comprehension between detailed (SHAP-based) and simplified security notifications, demonstrating that explainable AI significantly improves both metrics.

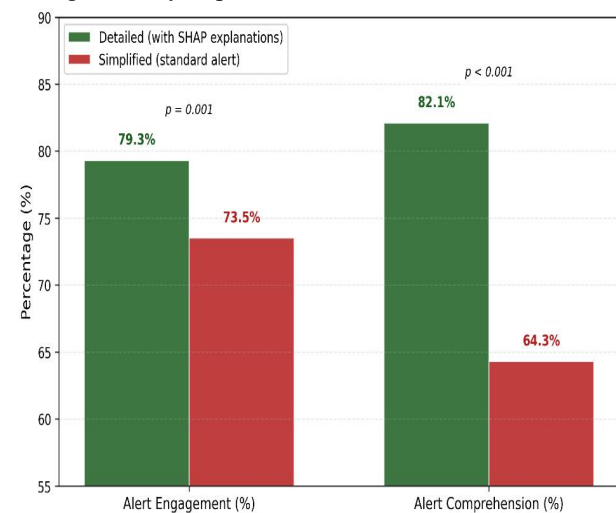


Figure 5: Impact of Explainable AI on User Engagement & Comprehension

DISCUSSION

The results prove the high effectiveness of light-weight machine learning-based cryptojacking detection on Android devices, while keeping the computational burden low. Random Forest outperformed all others with a high level of accuracy (F1_score=0.969), low computational burden (3.2 MB, 12.4 ms, 2.1% battery), and is robust in terms of time (−5.4% accuracy loss over 18 months) thus confirming H1 and H2. It validates that combining behavioural and network-based feature engineering significantly enhances detection. Features related to CPU variance, battery drain,

mining pool communication and cryptographic API usage were most discriminative, which is consistent with previous studies that showed cryptojacking has runtime behavioural signatures that are more reliable than static characteristics observed in (Hasan et al., 2025; Tekiner et al., 2022).

Our work emphasises computational sustainability for emerging economies. Though the accuracy of deep learning models was higher, it comes with increased latency (14.6–16.3 ms) and battery overhead (3.2–3.8%) than that of Random Forest (12.4 ms, 2.1%). This makes ensemble methods better for regions with inadequate charging infrastructure and hardware with low end specifications.

The concept drift analysis revealed that the ensemble models demonstrated higher temporal robustness compared to deep neural architectures, which may attribute to their ability to capture more general feature patterns instead of temporal noises. It is recommended that periodic retraining occurs to maintain the long-term effectiveness (for example quarterly).

The field deployment achieved an 87.7% infection reduction, demonstrating real-world impact. Importantly, the content of the XAI significantly increased user engagement (79.3% vs. 73.5%), comprehension (82.1% vs. 64.3%) and decreased user dismissals (15.8% vs. 20.6%) which supports H3. These findings are in line with Basheer et al. (2024), but add the dimension of actually measuring user behavior in a low-literate setting.

Comparison with Prior Cryptojacking Detection Works

We got $F1=0.969$ with our Random Forest model, which is comparable with Tekiner et al., 2022 (99% accuracy on top IoT networks) and Russo et al., 2021 (98.94% detection rate on NetFlow data). But unlike the previous studies, we explicitly study resource constraints (such as Battery, Latency, and Model size) and offer real-world field validation with user-centric XAI, which are not studied in previous work. Makkar et al. (2025) reported 98.27% accuracy and used extremely small inference time (0.000004 s) on static features, which was not used for cryptojacking or providing explanation for the user. Hence, our study has been filling a gap that integrates high accuracy, resource utilization and also practical deployability in emerging economies.

Limitations

Several limitations exist. First of all, the data set contains a variety of African malware samples, but other developing regions might have a varying threat landscape. Secondly, the study was conducted in

urban settings, with moderate literacy around smartphones, where results cannot be extrapolated to rural settings. Thirdly, concept drift analysis spanned 18 months of evaluation, but sustainability should be evaluated over a longer period.

Future studies could study Federated Learning in different Asian and Latin America regions and on Privacy - presenting Threat Intelligence and Adversarial Robustness. An additional improvement of user interaction might be possible through culturally adaptive XAI interfaces.

CONCLUSION AND IMPLICATIONS

This work introduced a lightweight AI based framework to detect cryptojacking on Android smartphones in emerging economies. We tested a set of 8,500 real-world applications from Africa, using a dataset that was rigorously validated, on multiple machine learning and deep learning methods in challenging deployments.

Both lightweight ensemble models, notably Random Forest, performed better: Low inference latency (12.4 ms), F1 score (0.969) Compact model size (3.2 MB), and low battery overhead (2.1%). These findings validate that capability to detect can be comparable to that of the more resource-consuming deep learning models, but without the need for costly hardware or constant access to the cloud.

When behavioral, network and cryptographic features were combined, the detection accuracy was significantly improved. Explainable AI mechanisms (SHAP) enhanced user engagement, understanding, trust, and showed that interpretable AI models can be effective even in low-digital-literacy environments. Translating the technical detection system into a practical local empowerment tool is increased when the users know what it is telling them.

Results from real-world deployment in Nigeria, Kenya, and South Africa demonstrated 87.7% reduction in rates of infection, validating laboratory-based models performed well in real-world threat environments. The findings of the temporal robustness analysis showed that lightweight ensemble methods were more robust against changing malware than the deep neural methods, thus leading to a decrease in the need for frequent and costly system overhauls. The work has practical applications, in the aspect of a scalable cyber security framework appropriate for emerging markets. The future approaches to mobile security should be not only accurate, but also energy efficient, explainable, cost-effective and lifelong maintainable. These features make it possible for communities to effectively adopt and maintain protections without relying on outside sources, making an external threat a manageable one.

Therefore, adopting lightweight AI-driven mobile threat detection is shown to be a viable solution to beef up the cybersecurity resilience of low-resource, distributed digital environments.

RECOMMENDATIONS

Based on the findings, several recommendations are proposed for various stakeholders.

Mobile security developers: To reduce battery life and inference delay in low-end devices like Android, lightweight ensemble models should be used over heavy deep-learning based architectures, to get a similar accuracy on detecting cryptojacking.

Emerging economies: Invest in the region's threat intelligence infrastructure (honey pots and partnerships with telecommunications companies) is strongly encouraged: models trained only on global threat feeds are between 2-3% less accurate at detecting locally sourced malware.

For cybersecurity educators and designers: It is suggested that principles of explainable AI (XAI) are incorporated in security awareness programs. Real-world effectiveness of transparent, contextualised alerts (82.1% user understanding, 79.3% engagement) outperformed the other populations (64.3% and 73.5% respectively).

For future works: Periodic retraining of the model (for example, retraining quarterly) should be considered to overcome concept drift, since even the proven model such as Random Forest exhibited a decrease in performance by 5.4 % after 18 months without updating.

For mobile network operators: Pricing lightweight AI-based security solutions with data plans in emerging markets is advised, with the 87.7% reduction in infections seen during field deployment, which could lead to reduced customer support expenses related to malware-compromised devices, increased user confidence and trust with their network, and better network performance reliability.

REFERENCES

- [1] Wanga H. Leveraging E-Commerce for Development: A Critical Review of Opportunities, Barriers and Strategies in Emerging Economies. *Int J Sci Res Mod Technol* 2026;120. <https://doi.org/10.38124/ijrsmt.v4i12.1094>.
- [2] Mothobi O, Kebotsamang K. The impact of network coverage on adoption of Fintech and financial inclusion in sub-Saharan Africa. *J Econ Struct* 2024;13:5. <https://doi.org/10.1186/s40008-023-00326-7>.
- [3] Mashoene MK, Schaling E. The Effect of Digital Financial Inclusion on Inclusive Growth and Poverty in Emerging and Developing Economies: A System-Generalized Method of Moments Model. *J Risk Financ Manag* 2026;19:236. <https://doi.org/10.3390/jrfm19040236>.
- [4] Ahmad AH, Green C, Jiang F. MOBILE MONEY, FINANCIAL INCLUSION AND DEVELOPMENT: A REVIEW WITH REFERENCE TO AFRICAN EXPERIENCE. *J Econ Surv* 2020;34:753–92. <https://doi.org/10.1111/joes.12372>.
- [5] Mannella L, Canavese D, Regano L. Detecting Cryptomining Traffic in IoT Networks. 2024 9th Int. Conf. Smart Sustain. Technol. Split., Bol and Split, Croatia: IEEE; 2024, p. 01–6. <https://doi.org/10.23919/SpliTech61897.2024.10612663>.
- [6] Russo M, Šrndić N, Laskov P. Detection of illicit cryptomining using network metadata 2021. <https://doi.org/10.21203/rs.3.rs-607598/v1>.
- [7] Almomani I, Alkhayer A, El-Shafai W. An Automated Vision-Based Deep Learning Model for Efficient Detection of Android Malware Attacks. *IEEE Access* 2022;10:2700–20. <https://doi.org/10.1109/ACCESS.2022.3140341>.
- [8] Tekiner E, Acar A, Uluagac S. A Lightweight IoT Cryptojacking Detection Mechanism in Heterogeneous Smart Home Networks. *Proc. 2022 Netw. Distrib. Syst. Secur. Symp., San Diego, CA, USA: Internet Society; 2022*. <https://doi.org/10.14722/ndss.2022.24208>.
- [9] Alshebli S, Mun H, Puthal D, Jamal Zemerly M, Martino L, Damiani E, et al. Enhanced Android Malware Detect Models Based on Explainable Generative Adversarial Networks. *IEEE Access* 2025;13:115898–908. <https://doi.org/10.1109/ACCESS.2025.3585241>.
- [10] Basheer N, Pranggono B, Islam S, Papastergiou S, Mouratidis H. Enhancing Malware Detection Through Machine Learning Using XAI with SHAP Framework. In: Maglogiannis I, Iliadis L, Macintyre J, Avlonitis M, Papaleonidas A, editors. *Artif. Intell. Appl. Innov.*, vol. 711, Cham: Springer Nature Switzerland; 2024, p. 316–29. https://doi.org/10.1007/978-3-031-63211-2_24.
- [11] Samuel JK, Jacob MT, Roy M, P M S, Joy AR. Intelligent Malware Detection System Based on Behavior Analysis in Cloud Computing Environment. 2023 Int. Conf. Circuit Power Comput. Technol. ICCPCT, Kollam, India: IEEE; 2023, p. 109–13. <https://doi.org/10.1109/ICCPCT58313.2023.10245065>.
- [12] Anisetti M, Ardagna CA, Bena N, Giandomenico V, Gianini G. Lightweight Behavior-Based Malware Detection. In: Chbeir R, Benslimane D, Zervakis M, Manolopoulos Y, Nguyen NT, Tekli J, editors. *Manag. Digit. Ecosyst.*, vol. 2022, Cham: Springer Nature Switzerland; 2024, p. 237–50. https://doi.org/10.1007/978-3-031-51643-6_17.
- [13] Ranpara R, Alsalman O, Kumar OP, Patel SK. A simulation-driven computational framework for adaptive energy-efficient optimization in machine learning-based intrusion detection systems. *Sci Rep* 2025;15:13376. <https://doi.org/10.1038/s41598-025-93254-4>.

- [14] Jaddoa A, Alharbi H, Hommadi A, Ismael HA. Toward Scalable and Sustainable Detection Systems: A Behavioural Taxonomy and Utility-Based Framework for Security Detection in IoT and IIoT. *IoT* 2025;6:62. <https://doi.org/10.3390/iot6040062>.
- [15] Singh Makkar I, Kumar Sinha A, Pratap T, A MK. Optimizing Android Malware Detection for Resource-Constrained Devices via a Two-Stage ML Pipeline. 2025 Int. Conf. Intell. Comput. Knowl. Extr. ICICKE, Bengaluru, India: IEEE; 2025, p. 1–6. <https://doi.org/10.1109/ICICKE65317.2025.11136486>.
- [16] Panman De Wit JS, Bucur D, Van Der Ham J. Dynamic Detection of Mobile Malware Using Smartphone Data and Machine Learning. *Digit Threats Res Pract* 2022;3:1–24. <https://doi.org/10.1145/3484246>.
- [17] Kim J, Ban Y, Ko E, Cho H, Yi JH. MAPAS: a practical deep learning-based android malware detection system. *Int J Inf Secur* 2022;21:725–38. <https://doi.org/10.1007/s10207-022-00579-6>.
- [18] Nasser AR, Hasan AM, Humaidi AJ. DL-AMDet: Deep learning-based malware detector for android. *Intell Syst Appl* 2024;21:200318. <https://doi.org/10.1016/j.iswa.2023.200318>.
- [19] Kaur A, Lal S, Goel S, Pandey M, Agarwal A. Android Malware Detection System using Machine Learning. *Proc. 2024 Sixt. Int. Conf. Contemp. Comput., Noida India: ACM; 2024, p. 186–91. https://doi.org/10.1145/3675888.3676049*.
- [20] Othman AMH, Ab Razak MF, @ Asmara SM, Ramli SN. Artificial Intelligence-Driven Detection of Android Malware Using Machine Learning Techniques. *Int J Adv Sci Eng Inf Technol* 2025;15:1656–62. <https://doi.org/10.18517/ijaseit.15.5.13405>.
- [21] Udayakumar P, Yalamati S, Mohan L, Haque MJ, Narkhede G, Bhashyam KM. Android malware detection using GIST based machine learning and deep learning techniques. *Indones J Electr Eng Comput Sci* 2024;35:1244. <https://doi.org/10.11591/ijeecs.v35.i2.pp1244-1252>.
- [22] Sawadogo Z, Mendy G, Dembele JM, Ouya S. Android malware detection: Investigating the impact of imbalanced data-sets on the performance of machine learning models. 2022 24th Int. Conf. Adv. Commun. Technol. ICACT, PyeongChang Kwangwoon_Do, Korea, Republic Of: IEEE; 2022, p. 435–41. <https://doi.org/10.23919/ICACT53585.2022.9728833>.
- [23] Hasan R, Biswas B, Samiun M, Saleh MA, Prabha M, Akter J, et al. Enhancing malware detection with feature selection and scaling techniques using machine learning models. *Sci Rep* 2025;15:9122. <https://doi.org/10.1038/s41598-025-93447-x>.
- [24] Prajwal HD, Zesheng C. Re-Evaluating Android Malware Detection: Tabular Features, Vision Models, and Ensembles. *Electronics* 2026;15:544. <https://doi.org/10.3390/electronics15030544>.
- [25] Rahman MR, Tohfa NA, Arif MH, Zareen S, Alim MA, Hossen MS, et al. Enhancing android mobile security through machine learning-based malware detection using behavioral system features. *World J Adv Res Rev* 2025;28:2308–15. <https://doi.org/10.30574/wjarr.2025.28.1.3686>.
- [26] Park S, Lee H, Kim D, Jun Moon H, Cho S-J, Hwang Y, et al. Enhancing the Sustainability of Machine Learning-Based Malware Detection Techniques for Android Applications. *IEEE Access* 2025;13:98876–87. <https://doi.org/10.1109/ACCESS.2025.3576733>.
- [27] Heena KausarS, Maria AnuV. Hybrid deep learning model for accurate and efficient android malware detection using DBN-GRU. *PLOS One* 2025;20:e0310230. <https://doi.org/10.1371/journal.pone.0310230>.
- [28] Parvathi S, Ponraj A, Parabrahmachari S, M A, Kumar MP, Krishnan VG. Android Malware Detection Using Deep Transfer Learning Model for IoT Applications. 2024 IEEE Int. Conf. Commun. Comput. Signal Process. IICCCS, ASANSOL, India: IEEE; 2024, p. 1–6. <https://doi.org/10.1109/IICCCS61609.2024.10763662>.
- [29] Nethala S, Chopra P, Kamaluddin K, Alam S, Alharbi S, Alsaffar M. A Deep Learning-Based Ensemble Framework for Robust Android Malware Detection. *IEEE Access* 2025;13:46673–96. <https://doi.org/10.1109/ACCESS.2025.3551152>.
- [30] Soi D, Sanna A, Maiorca D, Giacinto G. Enhancing android malware detection explainability through function call graph APIs. *J Inf Secur Appl* 2024;80:103691. <https://doi.org/10.1016/j.jisa.2023.103691>.
- [31] Yang J, Zhang Z, Zhang H, Fan J. Android malware detection method based on highly distinguishable static features and DenseNet. *PLOS ONE* 2022;17:e0276332. <https://doi.org/10.1371/journal.pone.0276332>.
- [32] García-Soto E, Martín A, Huertas-Tato J, Camacho D. Android Malware Detection Through a Pre-trained Model for Code Understanding. In: Bravo J, Ochoa S, Favela J, editors. *Proc. Int. Conf. Ubiquitous Comput. Ambient Intell. UCAmI* 2022, vol. 594, Cham: Springer International Publishing; 2023, p. 1055–60. https://doi.org/10.1007/978-3-031-21333-5_105.
- [33] Almarshad FA, Zakariah M, Gashgari GA, Aldakheel EA, Alzahrani AIA. Detection of Android Malware Using Machine Learning and Siamese Shot Learning Technique for Security. *IEEE Access* 2023;11:127697–714. <https://doi.org/10.1109/ACCESS.2023.3331739>.