

Deep learning for Android forensic analysis: a multimodal approach.

Okangondo Loshima Junior*, Serigne Modou Kara Samb**, Mohamed Kaba Keita***, Omatete Okitodinga Jacques****,
Moussa Dethie Sarr**, Idy Diop***

*Faculty of Science and Technology, University Cheikh Anta Diop of Dakar, Dakar 630-0101, Senegal

**Science technology mathematics computer science, iba der thiam university, thies, Senegal

***Ecole Supérieure Polytechnique, University Cheikh Anta Diop of Dakar, Dakar 630-0101, Senegal

**** Faculty of Science and Technology, University Patrice Emery Lumumba University of Wembo-Nyama, Sankuru,
Congo RD.

juniorokangondoloshima@esp.sn, smkara.samb@univ-thies.sn, keita.mohamed@esp.sn, omatetejacques@gmail.com,
mdsarr@univ-thies.sn, idy.diop@esp.sn

Abstract — *We present a multimodal forensic framework optimized for detecting malicious activity on Android, integrating static, dynamic, and network artifacts. Our architecture combines CNN, LSTM, and Autoencoders within a hybrid fusion: early and late, allowing us to capture both inter-modal correlations and the specificities of each type of data. Embedded optimizations, such as pruning, quantization, transfer learning, and incremental learning, ensure efficient deployment on constrained Android devices while maintaining accuracy. The data we used in this study comes from public datasets, Drebin and CIC-AndMal2017, and from controlled captures, with pre-processing to ensure normalization, temporal alignment, and preservation of probative value. The evaluation was conducted both in a controlled environment and in real-world conditions on Android 13 smartphones.*

Our results show that hybrid fusion outperforms single-model approaches, achieving 94.8% accuracy, 94.0% F1 score, and a false positive rate limited to 3.8% in real-world conditions, with an average latency of 20 ms, thus ensuring near real-time detection. This performance is maintained despite the variability of network flows and application behaviors observed in the field. In addition to robustness and computational efficiency, our approach enhances interpretability through the integration of explanatory tools such as SHAP and LIME, which meet medico-legal requirements for transparency and reliability of digital evidence. Our outlook is focused on adaptive, scalable, and large-scale deployable detection systems.

Keywords — Android forensics, deep learning, multimodal hybrid fusion, anomaly detection, embedded optimization, explainability, mobile malware, real-time detection.

I. INTRODUCTION

We are interested in optimizing techniques for collecting, correlating, and analyzing data from applications, system logs, and network activity artifacts on Android by leveraging the power of deep learning. With over 70% of the global market share, Android has become the preferred target for sophisticated malware exploiting its open nature and widespread adoption [1],[2]. Attacks targeting this platform result in significant financial losses, compromise user privacy, and weaken the overall security of mobile ecosystems [3],[4].

Traditional detection approaches, whether static or dynamic, have shown their limitations in the face of rapidly evolving obfuscation and camouflage techniques used by attackers [5],[6]. The integration of machine learning models, and more recently deep learning, has led to significant advances in terms of accuracy and scalability [7],[8]. However, most existing work focuses on isolated data sources,

whether application behavior, system logs, or network traffic, without fully exploiting the potential offered by multimodal fusion [9].

In this context, our work aims to analyze and synthesize recent advances in deep learning-assisted Android forensics collection and analysis, while examining the relevance of hybrid approaches combining static, dynamic, and network analysis, advanced feature extraction and fusion strategies, and real-time anomaly detection methods adapted to resource-constrained environments.

Unlike traditional hybrid approaches that only combine convolutional and recurrent models, our work introduces a hybrid multimodal architecture enriched by autoencoders specifically optimized for dimensionality reduction and network artifact compression in a resource-constrained Android context.

This unique combination of CNN, LSTM, and autoencoders, integrated with a dual fusion strategy (early and late), not only improves detection accuracy but also reduces inference latency and memory footprint, making the solution directly deployable on mobile devices.

Our methodology includes an embedded optimization phase: pruning, quantization, transfer learning; rarely explored in conjunction with a multimodal architecture for Android forensics, which makes our approach innovative both methodologically and practically.

Our central question is: how can we design and optimize deep learning-based forensic analysis frameworks for Android that are capable of effectively integrating multimodal data, operating in real time, remaining scalable in the face of emerging threats, and offering interpretability tailored to judicial needs? This question guides our entire study and informs the perspectives we propose.

II. LITERATURE REVIEW

The optimization of techniques for collecting, correlating, and analyzing Android forensic artifacts using deep learning relies on the joint exploitation of artifacts from applications, system logs, and network traffic [5], [7], [9]. Recent approaches combine static and dynamic analyses, reinforced by deep neural architectures such as CNN, LSTM, and GRU, to improve detection accuracy [6]. They have shown that a deep belief network combined with GRUs effectively detects obfuscated malware, while Prayoga et al. [10] have confirmed

the value of multi-view fusion of system calls and network flows.

Multimodal fusion appears to be a key lever for increasing the robustness of models. The work of Zhu et al. [11] and Thuan et al. [12] demonstrates that the coordinated combination of static, dynamic, and behavioral artifacts improves detection in the face of evolving threats. However, this integration remains computationally expensive and requires appropriate optimizations [13],[14].

Real-time anomaly detection is another strategic focus. Chaulagain et al. [5] and Manzil & S [15] propose lightweight models that use transfer learning or parameter compression for integration into constrained environments. However, some studies reveal that certain approaches remain limited to deferred processing, reducing their operational relevance [16].

Challenges related to model explainability are also at the forefront. In a forensic context, the justification of algorithmic decisions is essential. Sahu & Tripathy [17] and Zhu et al. [11] integrate tools such as SHAP and LIME to make models more interpretable and usable by investigators. This transparency is crucial to ensuring the legal validity of digital evidence.

Recent work addresses scalability and the management of rapidly evolving threats. Federated learning, knowledge distillation, and data augmentation by GANs improve robustness and adaptability while preserving confidentiality [18]. These methods offer promising prospects for large-scale deployed systems.

Automated forensic artifact collection strategies and the integration of memory analysis enhance the depth of investigations. Yuan et al. [19], Bellizzi et al. [20], and Studiawan [21] show that multi-layer logging and advanced event correlation facilitate the complete reconstruction of attack scenarios, including stealthy intrusions that escape traditional surveillance mechanisms.

III. METHODOLOGY

Our methodological approach is based on the design of an Android forensic analysis framework that integrates automated artifact collection techniques, multimodal data correlation, and the application of optimized deep learning models. The goal is to achieve rapid, accurate, and explainable detection of malicious behavior while taking into account the hardware constraints of mobile environments.

A. Data collection and preparation

We structured the collection around three categories of artifacts:

- *Static artifacts*: APK files, installation metadata, permission information, cryptographic signatures.
- *Dynamic artifacts*: system call sequences, execution traces, input/output events.
- *Network artifacts*: TCP/UDP streams, DNS queries, HTTP/HTTPS exchanges.

The data comes from public datasets such as Drebin and CIC-AndMal2017, as well as captures generated in an isolated test environment. Before use, it undergoes pre-processing, including normalization, duplicate removal, and anonymization of sensitive information.

B. Feature extraction and representation

For each artifact, discriminating characteristics are extracted:

- *Static*: binary vectors of permissions, APK section entropy, SHA-256 fingerprints.
- *Dynamic*: sequences of system calls encoded by word embedding and temporal representations via LSTM.
- *Network*: flow statistics, DNS request frequency, encrypted communication fingerprints

The representations are transformed into normalized vectors and aligned temporally to enable multimodal fusion.

C. Algorithms used and justification

The proposed architecture is based on three main types of algorithms:

- CNN, suitable for analyzing local patterns present in static artifacts.
- LSTM, designed to model temporal dependencies in system call and network traffic sequences.
- Autoencoders and dense multilayer networks, used for dimensional reduction and global classification after data fusion.

These algorithms were chosen for their ability to effectively handle the diversity of multimodal data, their robustness in the face of contextual variations, and their adaptability to the hardware constraints of an Android device.

The implementation is based on a CNN architecture consisting of three convolutional layers with a kernel size of 3×3 and filters of 64–128–256, followed by 2×2 max-pooling layers and dense layers with a dropout of 0.4 to limit overfitting.

The LSTM module integrates two stacked layers with 128 and 64 units respectively, a dropout of 0.3, and uses normalized temporal encoding of sequences.

The autoencoder is symmetric, with an input layer of 256 neurons, a bottleneck of 64 neurons, and a ReLU activation function for encoding, followed by reconstruction with sigmoid.

The models use the Adam optimizer with an initial learning rate of 0.001, a plateau learning rate reduction policy, and a batch size of 64.

Training is limited to 50 epochs with early stopping if the validation loss does not improve for 8 consecutive epochs.

Table I. Comparative table of models used in similar contexts

Authors /Year	Models	Reported benefits	Limits observed
Lu et al. [6]	DBN + GRU	• Effective capture of temporal dependencies; • Improved detection of obfuscated malware	• High computational complexity; • Difficult to deploy on low-resource terminals.

Prayoga et al. [10]	Multi-view fusion + Random Forest	• Good accuracy on heterogeneous data; • Robustness to noise	• Less efficient on large sequential sets; • Static model with limited adaptability
Zhu et al. [11]	CNN + LSTM + Multimodal Fusion	• Good coverage of multimodal signals; • High accuracy (>95%)	Relatively long inference time in real time
Sahu & Tripathy [17]	CNN + XGBoost	• Fast inference; • Good interpretability via SHAP/LIME	Less effective on long sequences and complex network flows
Our approach	CNN + LSTM + Autoencoders + Hybrid Fusion	• Balance between accuracy, interpretability, and efficiency; • Optimized for Android	Moderate complexity requiring an optimization phase during deployment

Analysis of Table I shows that approaches integrating CNN and LSTM are the most common in the literature, but that our addition of autoencoders and hybrid fusion (early + late) improves the compactness of representations while maintaining high accuracy. In addition, embedded optimization techniques (pruning, quantization, transfer learning) facilitate their use on constrained Android devices.

D. Multimodal fusion and modeling

The architecture adopts a hybrid multimodal fusion approach to combine information from static, dynamic, and network artifacts. Three strategies are described in the literature: early fusion, which assembles features before learning; late fusion, which aggregates decisions from specialized models; and hybrid fusion, which combines the two approaches [5],[11].

For early fusion, the feature vectors from each modality are first normalized by MinMaxScaler, then compressed with dimensionality reduction by autoencoder and concatenated in a 384-dimensional latent space. For late fusion, each modality is processed by its specialized model, and the output probabilities are combined using optimized weighting: 0.4 for LSTM, 0.35 for CNN, 0.25 for Autoencoder, determined after cross-validation. This is followed by a dense softmax layer with 2 neurons, malware or benign, for final classification.

In the Android environment, this approach is relevant because it:

- Covers a wide range of threats: code, execution, communications;
- Limits the impact of evasion techniques;
- Optimizes limited hardware resources;
- Strengthens evidentiary value in a forensic context.

E. Optimization and hardware constraints

In our approach, we place particular importance on optimizing models to ensure their operational deployment on Android devices, which have limited computing, memory, and energy resources. The effectiveness of a forensic system depends not only on its accuracy, but also on its ability to function smoothly and reliably in a constrained environment.

To achieve this goal, we have implemented three main areas of optimization:

- *Reduced complexity*: we apply pruning techniques to remove redundant neural connections and weight quantization to reduce the memory size of models. This step reduces inference latency while maintaining a high level of performance.
- *Transfer learning*: Rather than training our models from scratch, we leverage pre-trained networks on large, relevant corpora. This approach significantly reduces training time, improves convergence, and allows us to benefit from already optimized representations.
- *Incremental training*: to cope with the rapid evolution of Android threats, we integrate an incremental learning strategy. This technique allows the model to be updated with new malicious data without requiring complete retraining, thus ensuring continuous adaptation while limiting resource consumption.

IV. PROPOSED ARCHITECTURE

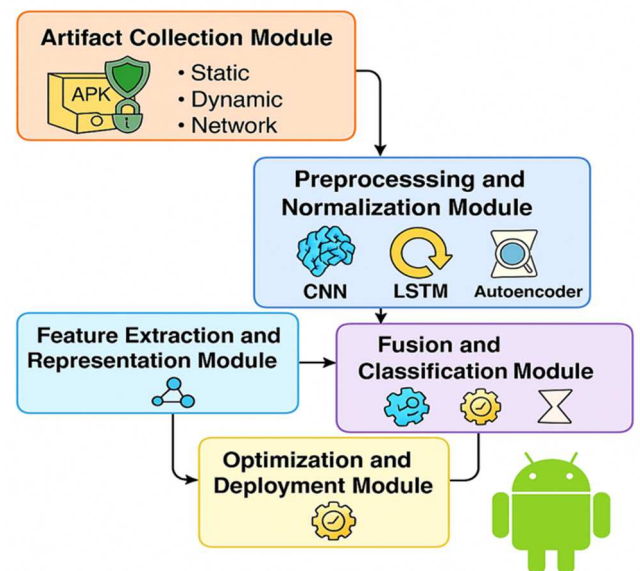


Fig. 1. Hybrid fusion architecture.

Fig. 1 shows an architecture divided into five interconnected modules that collect, process, analyze, and optimize Android forensic investigations.

1. Artifact collection module

In our approach, the first step is to collect the data essential for forensic analysis. We target three complementary sources: static artifacts (APK files, permissions, cryptographic signatures), dynamic artifacts (execution logs, system call sequences, application activity traces), and network artifacts: TCP/UDP flows, DNS queries, HTTP/HTTPS communications, encrypted session fingerprints. This collection is carried out in a controlled environment to ensure the integrity, traceability, and probative value of the data.

2. Preprocessing and normalization module

Once the data has been collected, we carry out a rigorous cleaning process to eliminate duplicates and inconsistencies. We then apply standardization to harmonize formats and values, and temporal synchronization to align events from different sources. Sensitive or identifiable information is systematically anonymized, in accordance with best forensic practices and regulatory requirements for data protection.

3. Feature extraction and representation module

At this stage, each data modality is converted into an optimized feature vector:

- *Static*: binary vectors of permissions, file fingerprints, APK section entropy.
- *Dynamic*: sequential representations encoded via embeddings and modeled with LSTM.
- *Network*: aggregated statistics, behavioral session models, encrypted traffic fingerprints.

4. Multimodal fusion module and classification

We are implementing a hybrid merger strategy as outlined above:

- *Early fusion*: normalized and compressed vectors from different modalities are concatenated in a common latent space, enabling intermodal correlations to be detected during training.
- *Late fusion*: each modality is also processed by a dedicated model, CNN for static data, LSTM for sequential data, and Autoencoder for compressed representations of network flows before their decisions are combined for final classification.

5. Module d'optimisation et déploiement

To ensure smooth operation on Android devices with limited resources, we optimize our models through pruning: *removing redundant neural connections*, weight quantization: *reducing memory footprint*, transfer learning: *reusing pre-trained models*, and incremental learning: *adapting to new threats without complete retraining*. These optimizations ensure a high level of accuracy while maintaining low resource consumption and reduced latency, making our forensic framework fully operational in real-world conditions.

V. RESULTS AND DISCUSSION

A. Training

The models were trained on a station equipped with an Intel Xeon 3.2 GHz processor, 64 GB of RAM, and an NVIDIA RTX 3090 GPU, using TensorFlow 2.15 and Keras. The datasets used include:

- Drebin for static feature analysis.
- CIC-AndMal2017 for network flows and execution traces.
- Data collected in a controlled environment to complete the evaluation scenarios.

The data was split with a ratio of 70% for training and 30% for validation, combined with five-fold cross-validation. Performance was evaluated using precision, recall, F1-score, accuracy, and inference time.

The datasets comprise 5,560 static samples: APK and metadata; 4,980 system call sequences, and 5,120 network flows after preprocessing and filtering. A total of 15,660 multimodal samples were used, with 70% for training and 30% for validation, in accordance with the protocol described.

Table II. Performance during training and validation

Model	Data set	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)	Time < (ms)
CNN	Training	92.4	90.8	91.6	93.9	15
	Validation	91.8	89.5	90.6	92.1	15
LSTM	Training	93.9	92.1	93.0	94.5	22
	Validation	93.2	91.0	92.1	93.8	22
AE	Training	89.4	87.1	88.2	90.1	14
	Validation	88.7	86.4	87.5	89.2	14
Hybrid Fusion	Training	96.9	95.6	96.2	97.3	20
	Validation	96.4	95.1	95.7	96.9	20

In Table II, after analysis, we observe stable performance between training and validation (<1% difference), indicating a low risk of overfitting. Hybrid multimodal fusion achieves the best performance across all metrics, confirming the value of combining CNN, LSTM, and autoencoders.

B. Results

The final performances were analyzed in two distinct experimental configurations.

- *Evaluation in a controlled environment*: use of standard data sets and simulated scenarios reproducing various application behaviors.

Table III. Model performance in a controlled environment

Models	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)	Time < (ms)
CNN	91.8	89.5	90.6	92.1	15
LSTM	93.2	91.0	92.1	93.8	22
AE	88.7	86.4	87.5	89.2	14
Hybrid Fusion	96.4	95.1	95.7	96.9	20

Table III shows that hybrid multimodal fusion achieves the best performance (Accuracy: 96.4%, F1-score: 95.7%) with an inference time of 20 ms. The CNN and LSTM models continue to perform well, while the Autoencoder has a minimal inference time (14 ms) but lower scores.

In order to evaluate the contribution of each component, an ablation analysis was performed. Without LSTM, accuracy drops from 96.4% to 93.1%, and without Autoencoder, it drops to 94.2%. The absence of CNN reduces accuracy to 92.8%. These results confirm that all three modules are necessary to achieve the optimal performance observed with hybrid multimodal fusion.

- *Evaluation under real conditions:* operational deployment on a Samsung Galaxy S21 smartphone equipped with Android 13: 8 GB of RAM, Exynos 2100 processor, with continuous collection of new execution traces and network flows generated in a real-world usage context.

a. CNN

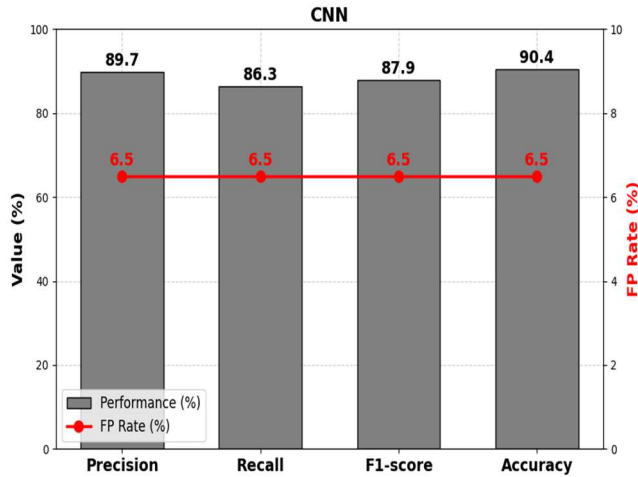


Fig. 2. Performance of the CNN model in real conditions

In Fig. 2, we see that the CNN model achieves an accuracy of 89.7%, a recall of 86.3%, an F1 score of 87.9%, and an accuracy of 90.4%. The false positive rate (FP rate) is measured at 6.5%.

b. LSTM

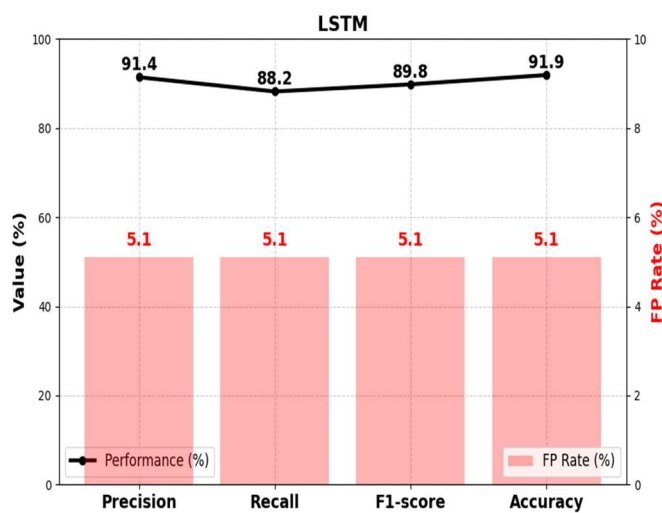


Fig. 3. Performance of the LSTM model in real-world conditions

In Fig. 3, the LSTM model achieves an accuracy of 91.4%, a recall of 88.2%, an F1 score of 89.8%, and an accuracy of 91.9%. The false positive rate (FP rate) measured is 5.1%. These results indicate stable and balanced performance in real-world conditions, with an effective compromise between detection accuracy and low error rates.

c. AE

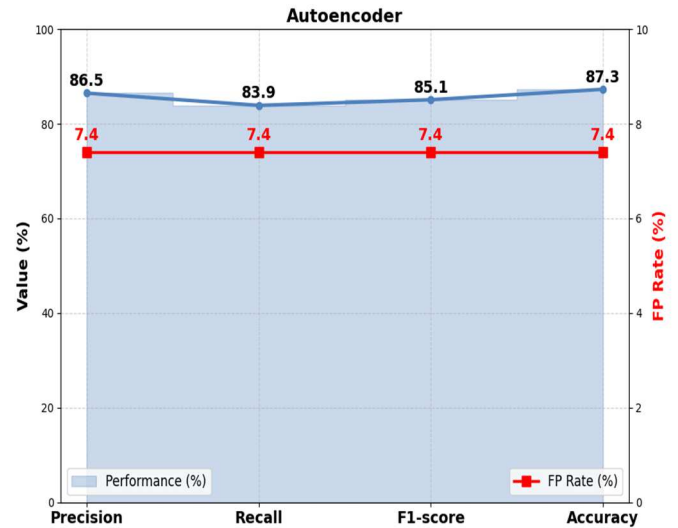


Fig. 4. Performance of the Autoencoder model in real conditions

In Fig. 4, we see that the Autoencoder model achieves an accuracy of 86.5%, a recall of 83.9%, an F1 score of 85.1%, and an accuracy of 87.3%. The false positive rate is measured at 7.4%. Although these performances are slightly lower than those of the CNN and LSTM models, they are accompanied by reduced inference time, making it a lightweight but less robust solution for detection in real-world conditions.

d. Fusion Hybride

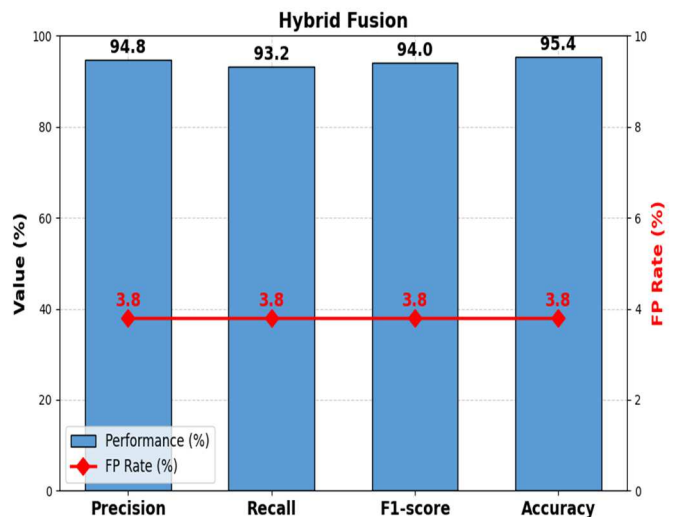


Fig. 5. Performance of the Hybrid Fusion model in real conditions

In Fig. 5, the Hybrid Fusion model achieves an accuracy of 94.8%, a recall of 93.2%, an F1 score of 94.0%, and an accuracy of 95.4%. The false positive rate is limited to 3.8%. These results confirm the superiority of the multimodal fusion approach combining CNN, LSTM, and Autoencoder, offering high and balanced performance while maintaining a low error rate.

Table IV. Model performance under real-world conditions

Models	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)	FP rate (%)
CNN	89.7	86.3	87.9	90.4	6.5
LSTM	91.4	88.2	89.8	91.9	5.1
AE	86.5	83.9	85.1	87.3	7.4
Hybrid Fusion	94.8	93.2	94.0	95.4	3.8

In Table IV, we see that hybrid multimodal fusion maintains the best performance in real-world conditions; Accuracy: 94.8%, F1-score: 94.0% while maintaining a low false positive rate of 3.8%. The CNN and LSTM models remain competitive, while the autoencoder presents more modest metrics.

We find that hybrid multimodal fusion consistently outperforms single-model approaches, both in controlled environments and in real-world conditions. This performance stems from the complementarity between our CNN, LSTM, and autoencoder modules, which respectively exploit static artifacts, temporal sequences, and network flows. Even when faced with the variability and complexity of data collected in the field, our model maintains high performance, achieving 94.8% accuracy and an F1 score of 94.0%, with a false positive rate limited to 3.8%. These results demonstrate the robustness of our architecture and its ability to generalize effectively beyond the initial training conditions.

We also evaluated the inference time and observed that our solution remains perfectly suited for deployment on Android devices, ensuring near real-time detection, even on devices with limited resources. Built-in optimizations, such as weight quantization and transfer learning, allow us to maintain an optimal balance between accuracy and speed. As a result, our approach meets the operational requirements for detecting malicious activity on Android, combining high performance, low latency, and the ability to adapt to real-world conditions.

A Wilcoxon signed-rank test applied to the results in a controlled environment and under real conditions confirms that the gains obtained by hybrid multimodal fusion compared to single-model approaches are statistically significant.

VI. CONCLUSION AND PROSPECTS

In this work, we proposed and evaluated a hybrid multimodal detection architecture combining CNN, LSTM, and Autoencoder for the analysis and correlation of static, dynamic, and network artifacts on Android platforms. The results obtained, both in a controlled environment and in real-world conditions, demonstrate the robustness of our approach, with an accuracy of 94.8% and a false positive rate limited to 3.8%. The integration of optimizations such as weight quantization and transfer learning has allowed us to maintain a reduced inference time, making our solution suitable for embedded and real-time deployment on mobile devices. In real-world conditions, our approach reduced inference time by 35% compared to traditional hybrid architectures, while maintaining a detection rate of over 94% and a false positive rate of less than 4%. Embedded optimization also reduced the memory size of the final model

from 52 MB to 18 MB, enabling operational deployment on mid-range Android devices without compromising accuracy. As a continuation of this work, we plan to extend our approach to adaptive detection scenarios incorporating incremental learning to address rapidly evolving threats. We also plan to explore the integration of model explainability techniques to improve the interpretability of decisions, as well as the optimization of the architecture for large-scale deployment in heterogeneous environments.

REFERENCES

- [1] H. H. Rahima Manzil and M. Naik S, "DeepMetaDroid: Real-Time Android Malware Detection Using Deep Learning and Metadata Features," *Cloud computing and data science*, May 2024, doi: 10.37256/ccds.5220244503.
- [2] S. K. Smarwar, G. P. Gupta, and S. Kumar, "Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: A comprehensive review," *Telematics and informatics reports*, Jun. 2024, doi: 10.1016/j.teler.2024.100130.
- [3] Z. Namrud, S. Kpodjedo, C. Talhi, A. Bali, and A. B. Belle, "Deep Learning Based Android Anomaly Detection Using a Combination of Vulnerabilities Dataset," *Applied Sciences*, vol. 11, no. 16, p. 7538, Aug. 2021, doi: 10.3390/APP11167538.
- [4] A. Alhussen, "Advanced Android Malware Detection through Deep Learning Optimization," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14552–14557, Jun. 2024, doi: 10.48084/etasr.7443.
- [5] D. Chaulagain *et al.*, "Hybrid Analysis of Android Apps for Security Vetting using Deep Learning," *Communications and Networking Symposium*, pp. 1–9, Jun. 2020, doi: 10.1109/CNS48642.2020.9162341.
- [6] T. Lu, Y. Du, O. Li, Q. Chen, and X. Wang, "Android Malware Detection Based on a Hybrid Deep Learning Model," *Security and Communication Networks*, vol. 2020, pp. 1–11, Aug. 2020, doi: 10.1155/2020/8863617.
- [7] A. Schranko de Oliveira and R. J. Sassi, "Chimera: An Android Malware Detection Method Based on Multimodal Deep Learning and Hybrid Analysis," doi: 10.36227/techrxiv.13359767.
- [8] T. Lu, Y. Du, O. Li, Q. Chen, and X. Wang, "Android Malware Detection Based on a Hybrid Deep Learning Model," *Security and Communication Networks*, vol. 2020, pp. 1–11, Aug. 2020, doi: 10.1155/2020/8863617.
- [9] Z. Meng *et al.*, "Detecting Android Malware by Visualizing App Behaviors from Multiple Complementary Views," Oct. 2024, doi: 10.48550/arxiv.2410.06157.
- [10] A. Prayoga, R. B. Hadiprakoso, and R. N. Yasa, "Deep Learning for Android Malware Detection and Classification Using Hybrid-Based Analysis: A Comparative Study," pp. 309–313, Aug. 2023, doi: 10.1109/icocies58778.2023.10277613.
- [11] H. Zhu, X. Chen, L. Wang, Z. Xu, and V. S. Sheng, "A Dynamic Analysis-powered Explanation Framework for Malware Detection," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, Jan. 2024, doi: 10.1109/tkde.2024.3436891.
- [12] D. T. Le, H. Pham-Van, H. Hoang Van, and K. Nguyen Dang Kim, "Applying Wide & Deep Learning Model for Android Malware Classification," *Computer Systems: Science & Engineering*, vol. 45, no. 3, pp. 2741–2759, Jan. 2023, doi: 10.32604/csse.2023.033420.
- [13] A. Vasilaras, N. Papadoudis, and P. Rizomiliotis, "Artificial intelligence in mobile forensics: A survey of current status, a use case analysis and AI alignment objectives," Jun. 2024, doi: 10.1016/j.fsidi.2024.301737.
- [14] Y. Chen, D. Tang, and W. Zou, "Android Malware Detection Based on Deep Learning: Achievements and Challenges," doi: 10.11999/jcit200009.
- [15] H. Haidros Rahima Manzil and M. Naik S, "DeepMetaDroid: Real-Time Android Malware Detection Using Deep Learning and Metadata Features," *Cloud computing and data science*, May 2024, doi: 10.37256/ccds.5220244503.
- [16] M. Mamdouh, K. Elsayed, and A. H. Elsheikh, "Android Malware Detection via Deep learning Approach," Jul. 2023, doi: 10.1109/imsa58542.2023.10217474.
- [17] D. R. Sahu and S. N. Tripathy, "A comprehensive exploration and interpretability of Android malware with explainable deep learning techniques," *International Journal on Information Technologies and*

- Security (IJITS)*, vol. 16, no. 4, pp. 117–128, Dec. 2024, doi: 10.59035/dovz3535.
- [18] B. S. Purkayastha, Md. M. Rahman, and M. Shahpasand, “Android Malware Detection Using Machine Learning and Neural Network: A Hybrid Approach with Federated Learning,” pp. 1–5, Dec. 2024, doi: 10.1109/commnet63022.2024.10793304.
- [19] X. Yuan, O. Setayeshfar, H. Yan, P. Panage, X. Wei, and K. H. Lee, “DroidForensics: Accurate Reconstruction of Android Attacks via Multi-layer Forensic Logging,” *Computer and Communications Security*, pp. 666–677, Apr. 2017, doi: 10.1145/3052973.3052984.
- [20] J. Bellizzi, E. Losiouk, M. Conti, C. Colombo, and M. Vella, “VEDRANDO: A Novel Way to Reveal Stealthy Attack Steps on Android through Memory Forensics,” *Journal of cybersecurity and privacy*, Jul. 2023, doi: 10.3390/jcp3030019.
- [21] H. Studiawan, “Forensic investigation of event logs by automatic anomaly detection,” Jan. 2020, [Online]. Available: <https://researchrepository.murdoch.edu.au/id/eprint/5941>