

Comparative evaluation of the performance and effectiveness of Machine Learning and Deep Learning algorithms for anomaly detection in Android log analysis.

Okangondo Loshima Junior*, Serigne Modou Kara Samb**, Mohamed Kaba Keita***, Doudou Fall***, 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

juniorokangondoloshima@esp.sn, smkara.samb@univ-thies.sn, keita.mohamed@esp.sn, doudou.fall@esp.sn, mdsarr@univ-thies.sn, idy.diop@esp.sn

Abstract - Due to the substantial rise in cyberattacks aimed at Android devices, log analysis has become critical for identifying and mitigating security anomalies. This article conducts a comparative analysis of machine learning and deep learning algorithms for the detection of anomalies within Android logs, specifically sourced from the LogHub dataset. These logs, gathered from Android smartphones equipped with extensive instrumentation, are both rare and intricate. The complexity arises from Android's multithreaded architecture, making anomaly detection a challenging task.

The aim of this study is to identify the most suitable algorithms for detecting anomalies in Android logs, evaluating three algorithms per category (machine learning and deep learning). Using metrics such as precision, recall, F1 score and AUC-ROC, the analysis highlights the effectiveness of each approach depending on context. Finally, recommendations are proposed for optimizing current methodologies and exploring new research perspectives.

Keywords: ML, DL, DBSCAN, CNN, RNN, SVM, OCSVM.

I. INTRODUCTION

The massive adoption of Android devices worldwide has led to a significant increase in cyberattacks targeting this operating system [1][2]. The logs generated by the Android system, which record essential information on applications, user activities and system performance, play a crucial role in monitoring and detecting anomalies [3]. However, the analysis of these logs has become increasingly complex due to their volume and unstructured nature, necessitating the use of sophisticated tools capable of processing and interpreting this data in an automated way.

The aim of this study is to compare the performance of machine learning and deep learning algorithms for analyzing Android logs, in order to determine which approaches are best suited to detecting anomalies in Android logs. Three algorithms representative of each method were evaluated: Isolation Forest, DBSCAN and One-Class SVM for machine learning; Autoencoder, CNN and RNN for deep learning. Using metrics such as precision, recall, F1-score and AUC-ROC, this comparison aims to identify the algorithms best suited to Android log analysis, and to offer recommendations for optimizing their use in cybersecurity.

II. JUSTIFICATION AND CHOICE OF ALGORITHMS

The choice of machine learning and deep learning algorithms for this study is based on their state-of-the-art relevance to the specific context of log analysis, and their ability to handle complex anomalies.

For machine learning, we have selected Isolation Forest, DBSCAN, and One-Class SVM, as these algorithms are particularly well suited to anomaly detection:

- Isolation Forest is commonly used to identify anomalies in large, unlabeled datasets, by isolating rare data points efficiently.
- DBSCAN is a robust clustering algorithm that detects anomalies by identifying data points that do not correspond to any density cluster.
- One-Class SVM is suitable for scenarios where only normal data is available for training, which corresponds well to the challenges posed by Android logs.

For deep learning, we opted for Autoencoders, CNNs, and RNNs (LSTM), each offering distinct advantages:

- Autoencoders are excellent for dimension reduction and anomaly identification by learning from compressed data representations.
- CNNs are known for their ability to extract complex local features, which is useful for detecting anomaly patterns in logs.
- RNNs (LSTMs) are particularly effective at capturing temporal dependencies, a crucial aspect in the analysis of Android logs that follow a chronological sequence.

These algorithms were chosen because they cover a wide range of techniques and meet the specific needs of anomaly detection in complex, massive data such as Android logs.

III. REVIEW OF LITERATURE

III.1. Machine learning algorithms

a) Isolation Forest

Isolation forests are increasingly used in log analysis for anomaly detection, thanks to their ability to identify new intrusion patterns and anomalies in log data. They have shown increased effectiveness, for example, by increasing the accuracy of intrusion detection systems by 300% when applied to http log data [4]. In software-defined networks, they effectively detect outliers, although they face challenges in terms of precision and recall [5]. In addition, they outperform traditional methods for monitoring vital signs in healthcare [6].

b) DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a powerful algorithm for log analysis, particularly effective in identifying anomalies in various fields thanks to its ability to detect clusters of varying densities. In the field of network security, DBSCAN has been used to efficiently identify anomalies while reducing false positives, thus improving the detection of security events [7]. Its application in the analysis of vehicle CAN logs, in conjunction with LSTM autoencoders, enabled significant pattern recognition, although the algorithm showed limitations when faced with minor signal modifications [8]. In the field of aeronautical data monitoring, DBSCAN has demonstrated its effectiveness in detecting outliers and predicting safety risks in real time [9].

c) One-Class SVM

The Single Class Support Vector Machine (OCSVM) is an effective method for analyzing logs, particularly for identifying anomalies in environments where labeled data is scarce, as it relies on normal log entries to detect deviations. For example, Panigrahi demonstrated that OCSVM could achieve 88% accuracy in identifying anomalies from logs from complex systems [10]. Similarly, Nguyen et al. showed that OCSVM improved operational efficiency by detecting anomalies in logs from Juniper routers [11].

III.2. Deep learning algorithms

a) Autoencoder

Log analysis using autoencoders is a deep learning technique that has proved effective in a variety of fields, such as geological exploration and systems monitoring. In oil exploration, autoencoders can be used to analyze well logs, improving the correlation of geological layers by integrating multiple log data [12]. In the context of system logs, these models detect anomalies using reconstruction errors and vectorization techniques such as bag-of-words, facilitating the conversion of log messages into a digital format suitable for analysis [13].

b) CNN

Log analysis using convolutional neural networks (CNNs) has emerged as a powerful technique for improving network security and operational efficiency, by enabling anomaly detection and failure prediction. For example, a 1D-CNN model combined with BERT has demonstrated superior performance in identifying network attacks through robust

anomaly detection in system logs [14]. In addition, the use of a BERT-CNN model to predict virtual machine failures showed notable effectiveness, with an F1 score of 0.74 [15]. In addition, the FA-CNN model, incorporating principal component analysis (PCA), has been proposed to improve the analysis of traffic logs, thus enhancing the effectiveness of network security systems [16].

c) RNN

Recurrent neural networks (RNNs) have become a powerful tool for log analysis, particularly effective in detecting anomalies in system logs due to their ability to process sequential data. For example, closed recurrent units (GRUs) integrated with attention mechanisms have shown increased efficiency in identifying anomalies in cloud computing logs by focusing on relevant parts of the data [17]. In addition, the combination of advanced natural language processing (NLP) techniques, such as BERT, with RNNs enriches the representations of log messages, improving the detection of unusual activity [18].

III.3. Discussion

Several approaches using machine learning and deep learning algorithms have been explored for log analysis, each presenting specific advantages and challenges. Isolation Forest showed an increased ability to detect anomalies in various contexts, notably in cybersecurity, although it encounters difficulties in terms of precision and recall, requiring further optimizations such as adjusting the contamination parameter, selecting relevant features, applying cross-validation, integrating regularization techniques, and filtering out false positives. DBSCAN, meanwhile, has proved effective in identifying anomaly clusters in network and aeronautical logs, but lacks finesse in detecting subtle anomalies, suggesting the need for complementary approaches.

One-Class SVM, particularly useful in environments with little labeled data, has demonstrated good accuracy in detecting anomalies in complex systems, but its effectiveness is highly dependent on feature selection. On the side of deep learning algorithms, autoencoders have proven their usefulness in log analysis by exploiting reconstruction errors to identify anomalies, although they are limited by class imbalance.

CNNs, with their robust anomaly detection capabilities, have established themselves as a powerful tool for improving network security, despite the challenges posed by complex log structures.

As for RNNs, thanks to their sequence processing, they are effective in detecting anomalies in cloud computing logs, but their performance can be affected by confidentiality issues and limitations in text vectorization. This work underlines the need for continuous adaptation of methods to optimize log analysis in a variety of environments.

IV. METHODOLOGY

IV.1. Collecting Android logs

a) Description of log sources on Android

Android logs are detailed records generated by the operating system and applications installed on the device. These logs capture various events such as user interactions,

errors, system events, and background operations. On Android, we have several sources of logs, here are the main ones [19]:

- *Logcat*: It is the main logging tool used by Android, recording messages from the operating system as well as from applications [20]. Logs generated by Logcat include information on running processes, exceptions, debugging, and custom messages from developers.
- *Event Logs*: stores system events related to security, performance and network activity. They are useful for monitoring anomalies and unusual behavior on the device.
- *System Logs*: Capture information on critical system operations, such as system start-up, crashes, and other events relevant to system diagnostics and maintenance.
- *Application-specific Logs*: These are logs generated by the applications installed on the device, and contain information specific to each application, such as user interactions, specific errors, and other data relevant to the application.

b) Log collection and export methods

Android logs can be collected in real time or a posteriori, depending on the needs of the analysis. Commonly used methods are [21]:

- *Using the ADB (Android Debug Bridge) tool*: is a command-line tool that allows users to connect to an Android device from a computer. ADB commands, such as "adb logcat", can be used to retrieve logs in real time or extract historical logs.
- *Python scripts*: scripts can be configured to run ADB commands, capture data and export it in structured formats such as CSV or JSON for easier analysis.
- *Third-party logging applications*: Some Android applications, for example: logcat reader, aLogcat, CatLog, SysLog, logcat extreme, etc. [20] are specifically designed to capture and export logs in a simplified way. These applications allow logs to be accessed directly from the device, without the need to connect via ADB.
- *LogHub* [22]: Is a platform that provides log datasets collected from various systems, including Android, a valuable source for obtaining real system logs. Logs provided by LogHub are already structured and ready for analysis, simplifying the collection process. We have used these logs for our analyses in this comparative study.

IV.2. Data pre-processing

a) Log cleaning and standardization techniques

It is essential to clean and normalize logs prior to any analysis to ensure that the data is consistent and ready for machine learning or Deep learning algorithms. Cleaning and normalization can be achieved by removing redundant or irrelevant data, managing values, normalizing date and time formats, converting categorical values that often contain categorical data (such as log levels: INFO, DEBUG,

ERROR) [23]. These values are converted into numerical or binary representations for use in machine learning models.

b) Extraction of features relevant to analysis

Feature extraction is a key step in data pre-processing, transforming raw logs into a format that algorithms can use effectively. For Android logs, the following features can be extracted:

- *Log message length*: Message length can indicate the complexity or severity of an event. For example, error messages are often longer and more detailed.
- *Log level*: Each log message is associated with a level (INFO, DEBUG, WARNING, ERROR). This level is a direct indicator of the importance or severity of the event and is often used as a feature for classification or anomaly detection.
- *Frequency of events*: The number of times a certain type of event occurs in a given time interval can be a useful characteristic. For example, a sudden increase in error messages may indicate a potential problem.
- *Time dependencies*: Sequences of events can reveal important temporal patterns. By analyzing temporal dependencies, algorithms can better understand the flow of event execution and detect anomalies that break these patterns.
- *Error contextualization*: The position of an error message in the context of other events is key to understanding the cause of an anomaly. Features can include the proximity of error messages to other event types.

III.3. Log Analysis with Machine Learning Algorithms

a) Description of algorithms and techniques used

The analysis of Android logs was carried out using three well-established machine learning algorithms, each with its own strengths in anomaly detection. The algorithms selected for this study are: Isolation Forest, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and One-Class SVM (Support Vector Machine).

1. *Isolation Forest*: This algorithm is based on the construction of binary decision trees that progressively isolate data points by dividing the feature space [24]. Data points that are isolated quickly (with few divisions) are considered anomalies. The advantage of Isolation Forest is its ability to handle large volumes of data and to detect anomalies non-parametrically.

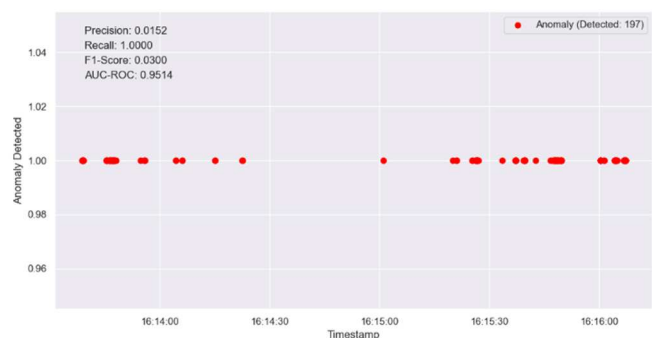


Fig. 1. Log analysis result for the Isolation Forest algorithm [34]

2. **DBSCAN**: A density-based clustering algorithm, DBSCAN identifies high-density clusters in the data and considers points outside these clusters as anomalies [25]. It is particularly effective at detecting anomalies that lie in low-density regions in feature space.

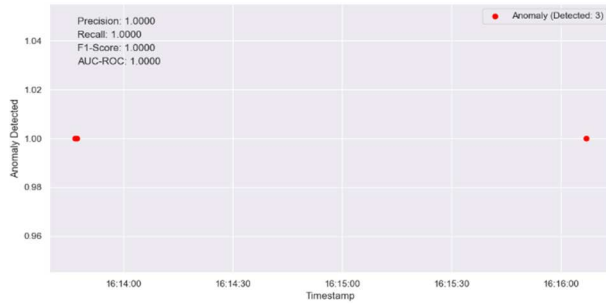


Fig. 2. DBSCAN algorithm log analysis result [34]

3. **One-Class SVM**: uses a kernel function to learn a boundary around normal data points, separating out anomalies. One-Class SVM is suitable for scenarios where anomalies are rare and difficult to model directly. [26]

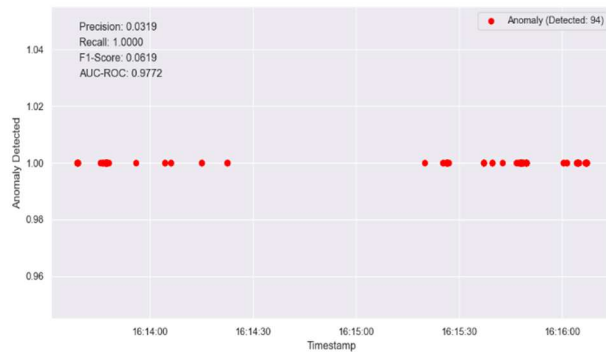


Fig. 3. One-Class SVM algorithm log analysis result [34]

b) Performance analysis and evaluation process

Each algorithm was applied after appropriate pre-processing and extraction of relevant features. The analysis steps were followed by a performance evaluation using several metrics common in anomaly detection:

- **Precision**: precision measures the proportion of points identified as anomalies that are actually anomalies. It is an essential measure for assessing the reliability of an algorithm in scenarios where false positives can lead to unnecessary alerts. [27]
- **Recall**: Recall measures the proportion of actual anomalies that have been correctly identified by the algorithm. High recall is crucial to minimize the risk of missing critical anomalies. [28]
- **F1 score**: The F1 score is the harmonic mean of precision and recall, providing a balanced overview of the algorithm's performance. [29]
- **AUC-ROC** (Area Under the Curve - Receiver Operating Characteristic): This metric evaluates an algorithm's ability to distinguish anomalies from normal data by plotting the rate of true positives

against the rate of false positives at different decision thresholds. [30]

III.4. Log Analysis with Deep Learning Tools

a) Description of algorithms and techniques used

Deep learning techniques applied to Android log analysis rely on the ability of models to learn complex representations of data, often with improved anomaly detection results compared to traditional machine learning algorithms. In this study, three main types of models were used: Autoencoders, Recurrent Neural Networks (RNN), and Convolutional Neural Networks (CNN).

1. **Autoencoders**: are neural networks that learn to compress input data into a lower-dimensional representation (encoding) and then reconstruct the original data from this representation. Anomalies are detected when the reconstruction is poor, i.e. when the difference between the input data and the reconstructed data exceeds a certain threshold. [31]

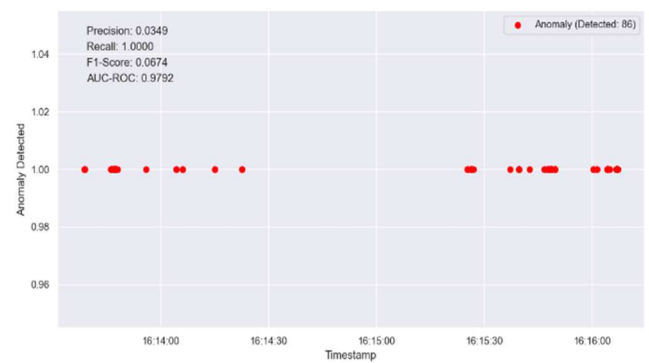


Fig. 4. Autoencoder algorithm log analysis result [34]

2. **Recurrent Neural Networks (RNNs)**: RNNs, and more specifically LSTMs (Long Short-Term Memory), are well suited to the analysis of temporal sequences, such as logs. LSTMs can capture temporal dependencies in data, which is essential for detecting anomalies that manifest themselves across sequences of events rather than in isolated data points. RNNs are able to "remember" information across multiple temporal steps, making them effective for modeling complex patterns in logs. [32]

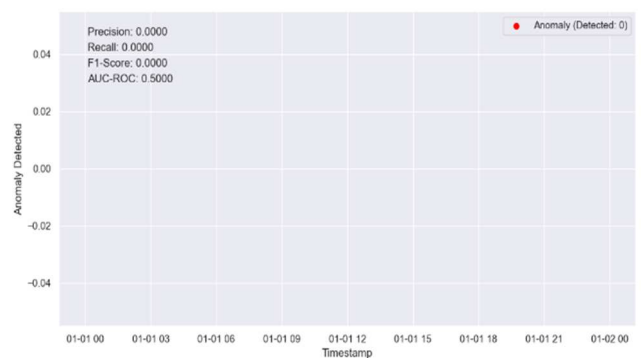


Fig. 5. Algorithm log analysis result RNN [34]

3. *Convolutional Neural Networks (CNN)*: traditionally used for image analysis, CNNs can be applied to data sequences such as logs. Convolutional layers extract local features from sequences, capturing complex patterns that may indicate anomalies. [33]

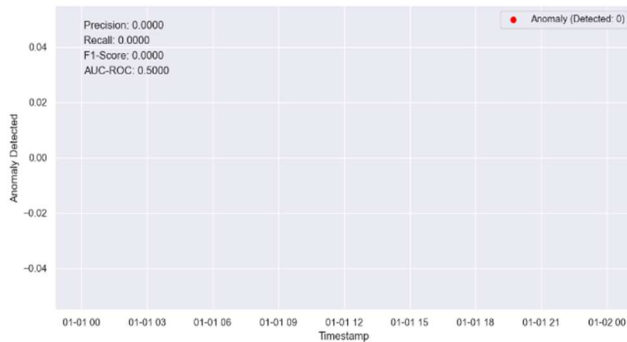


Fig. 6. Algorithm log analysis result CNN [34].

b) Performance analysis and evaluation process

Deep learning models were trained on Android logs after appropriate normalization and transformation of the data into temporal sequences. Model performance was evaluated using metrics similar to those for machine learning algorithms, with a particular focus on the models' ability to detect anomalies in complex sequences.

- *Mean Squared Error (MSE)*: Used mainly with autoencoders, MSE measures the difference between input data and reconstructed data. A threshold is set to identify anomalies, and the accuracy of the model is evaluated according to this threshold.
- *Accuracy and Recall*: As with machine learning algorithms, these metrics are used to assess the ability of Deep learning models to correctly identify anomalies without generating too many false positives.
- *ROC and AUC curves*: These curves are also relevant for assessing the ability of Deep learning models to discriminate anomalies from normal data. They enable the overall performance of the model to be visualized at different decision thresholds.

V. RESULTS, CHALLENGES AND DISCUSSIONS

V. 1. Results

TABLE I. Summary table of results obtained after analysis

Algorithms	Precision	Recall	F1-Score	AUC-ROC	Anomalies
Machine learning					
Isolation Forest	0,00152	1,0000	0,0300	0,9514	197
DBSCAN	1,0000	1,0000	1,0000	1,0000	3
One-Classe SVM	0,0319	1,0000	0,0619	0,9772	94
Deep learning					

CNN	0,0000	0,0000	0,0000	0,5000	0
Autoencoder	0,0349	1,0000	0,0674	0,9792	86
RNN (LSTM)	0,0000	0,0000	0,0000	0,5000	0

Fig. 6. Summary table of results obtained in each algorithm

In this table summarizing the results, the analysis of the performance of machine learning (ML) and deep learning (DL) algorithms applied to anomaly detection in Android logs revealed notable discrepancies in terms of accuracy and number of anomalies detected. Isolation Forest detected a total of 197 anomalies, distinguishing itself by its effectiveness in identifying a wide range of anomalies. However, this performance is mitigated by an extremely high false positive rate, reflected in a precision of just 0.0152. Although recall was perfect (1.0000), indicating that all true anomalies were captured, the very low F1 score of 0.0300 highlights a high proportion of incorrect detections.

DBSCAN, on the other hand, identified 3 anomalies, with perfect precision and recall, resulting in an equally perfect F1 score of 1.0000. These results suggest that DBSCAN is extremely accurate when detecting anomalies, but its ability to identify a larger number is limited, which could restrict its usefulness in scenarios where wider coverage is required.

The One-Class SVM detected 94 anomalies, with a precision of 0.0319 and a perfect recall of 1.0000. The F1 score of 0.0619 reveals a moderate performance, largely influenced by a substantial number of false positives, which limits its overall effectiveness despite its ability to capture all the anomalies present.

Concerning the deep learning algorithms, Autoencoders identified 86 anomalies, achieving a precision of 0.0349 and a recall of 1.0000. However, the F1 score of 0.0674 indicates that the algorithm is also penalized by a high false positive rate, despite good anomaly coverage.

RNN (LSTM) and CNN detected no anomalies, resulting in zero scores for all relevant metrics, suggesting that these algorithms, in their current configuration, are not suitable for this type of Android log analysis.

V. 2. Challenge and discussion

The results of this study reveal marked differences between algorithms for detecting anomalies in Android logs. Isolation Forest, while effective in detecting many anomalies, is limited by low accuracy due to a high false positive rate. DBSCAN stands out for its perfect accuracy, but its low number of detections may limit its application to scenarios where more exhaustive detection is required.

One-Class SVM, despite its ability to identify all anomalies, also suffers from a high rate of false positives, requiring improved feature selection to maximize its effectiveness. Autoencoders show promising potential, but their sensitivity to class imbalances reduces their applicability, while RNN and CNN failed to detect any anomalies, suggesting that they require retraining or adaptation of parameters such as sequence size, number of layers and neurons, activation function, learning rate, batch size, and the application of regularization techniques to better address the specifics of Android logs.

Future work could focus on optimizing deep learning algorithms for Android log analysis, notably by adjusting critical hyperparameters such as learning rate, batch size, number of layers and neurons, and activation function. These adjustments would aim to reduce false negatives and improve model sensitivity. In addition, the integration of regularization techniques such as Dropout or L2 penalization could be explored to minimize overlearning and improve model robustness. Furthermore, the integration of hybrid methods combining the strengths of machine learning and deep learning algorithms could offer a more robust solution for anomaly detection, maximizing accuracy while minimizing false positives. Finally, the study of new neural network architectures, such as Transformer-based models or specialized architectures for log sequence processing, could open up new avenues for improving the performance of deep learning approaches in this field.

CONCLUSION

The results of this study highlight the marked differences between machine learning and deep learning algorithms in terms of their effectiveness in detecting anomalies in Android logs. Machine learning algorithms, in particular DBSCAN, proved more reliable in terms of accuracy, although their ability to detect a wide range of anomalies is limited. Isolation Forest, although capable of detecting a large number of anomalies, has too high a false positive rate for sensitive applications. Deep learning algorithms, while promising, still require significant fine-tuning to achieve performance comparable to that of machine learning algorithms in this specific context.

By way of conclusion, the choice of the most suitable algorithm for Android log analysis will depend heavily on the application context and specific anomaly detection requirements. Machine learning algorithms, such as DBSCAN, could be preferred for scenarios requiring high accuracy, while Isolation Forest could be considered for contexts where broad coverage is required, subject to additional adjustments such as reducing the false positive rate by adjusting the contamination parameter, improving feature selection, and increasing tree depth. Deep learning algorithms still require improvement to be competitive in this specific field.

REFERENCES

- [1] A. Khatoon and P. Corcoran, "Android permission system and user privacy — A review of concept and approaches," 2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin), Berlin, Germany, 2017, pp. 153-158, doi: 10.1109/ICCE-Berlin.2017.8210616.
- [2] G. Altan and F. Pasalioglu, "Android Malware Application Detection using Multi-layer Perceptron," *Journal of Intelligent Systems and Applications*, pp. 95-99, 2022, doi: 10.54856/jiswa.202212221.
- [3] J. M. R. Bessa, R. L. Z. Filho, G. L. de Souza Filho, L. Pessoa, R. Barreto, and R. de Freitas, "Performance Analysis of Log Parsers from the Logpai Collection on Raw Data from Android Devices," 2024, doi: 10.5753/wperformance.2024.2423.
- [4] D. Karev, C. McCubbin, and R. Vaulin, "Cyber Threat Hunting Through the Use of an Isolation Forest," in *Proceedings of the 2017 International Conference on Data Intelligence and Security (ICDIS)*, 2017, pp. 163-170, doi: 10.1145/3134302.3134319.
- [5] M. S. Lakshmi et al., "Evaluating the Isolation Forest Method for Anomaly Detection in Software-Defined Networking Security," *Deleted Journal*, vol. 19, no. 4, pp. 279-297, 2024, doi: 10.52783/jes.639.
- [6] K. Yadav, S. P. Dwivedi, U. Singh, S. N., V. Saravanan, and N. Kumar, "Isolation Forest Anomaly Detection in Vital Sign Monitoring for Healthcare," in *Proceedings of the 2023 IEEE International Conference on Artificial Intelligence and Human-Computer Interaction (ICAHI)*, 2023, pp. 1-7, doi: 10.1109/icaihi57871.2023.10488940.
- [7] W. Yi, N. Yin, L. Zhe, X. Dong, et J. Pan, "Network Abnormal Log Clustering Algorithm Based on DBSCAN", in *2023 IEEE/CIC International Conference on Communications in China (ICCC)*, 2023, pp. 42-47, doi: 10.1109/ICC59986.2023.10420855.
- [8] Chandralekha, C. H M, N. P S, S. P. S and M. K. Ghosh, "Anomaly detection in recorded CAN log using DBSCAN and LSTM Autoencoder," *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*, Bangalore, India, 2022, pp. 1-7, doi: 10.1109/GCAT55367.2022.9971885.
- [9] U. R. Omonov, "Analysis of Abnormal Flight and Controllers Data Based on DBSCAN Method", *Security and Communication Networks*, vol. 2022, pp. 1-8, 2022, doi: 10.1155/2022/7474270.
- [10] D. A. Bhanage and A. V. Pawar, "Improving Classification-Based Log Analysis Using Vectorization Techniques," in *Proceedings of Third International Conference on Advances in Computer Engineering and Communication Systems*, A. B. Reddy, S. Nagini, V. E. Balas, and K. S. Raju, Eds., vol. 612, *Lecture Notes in Networks and Systems*, Singapore: Springer, 2023, pp. 1-7. doi: 10.1007/978-981-19-9228-5_24.
- [11] Y. Liu, J. Lv, S. Ma, and W. Yao, "The Runtime System Problem Identification Method Based on Log Analysis," in *Proceedings of the 2018 International Conference on Computer Communication and Networks (ICCCN)*, 2018, pp. 1-7. doi: 10.1109/ICCCN.2018.8487466.
- [12] J. Fu, R. Li, T. Gao, P. Zhang, et C. Wang, "Deep Learning Applications for Well Log Correlation," *2024 International Petroleum Technology Conference (IPTC)*, doi: 10.2523/iptc-24062-ms.
- [13] Á. Puskás, E. Kail, S. Laczi, et A. Banati, "Neural Network-Based Log Analysis Methods for 5G Network," in *2023 IEEE 21st International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 589-594, 2023, doi: 10.1109/SISY60376.2023.10417905.
- [14] H. P. Pham, V. H. Nguyen, and N. A. Tran, "Log Analysis for Network Attack Detection Using Deep Learning Models," 2023, doi: 10.1145/3628797.3628943.
- [15] S. Nam, J. -H. Yoo and J. W. -K. Hong, "VM Failure Prediction with Log Analysis using BERT-CNN Model," *2022 18th International Conference on Network and Service Management (CNSM)*, Thessaloniki, Greece, 2022, pp. 331-337, doi: 10.23919/CNSM55787.2022.9965187.
- [16] D. Wu, Y. Tian, W. Chen, Y. Tan, R. Zhang and X. Yang, "Traffic Log Analysis Model based on FA-CNN," *2023 Panda Forum on Power and Energy (PandaFPE)*, Chengdu, China, 2023, pp. 148-153, doi: 10.1109/PandaFPE57779.2023.10141222.
- [17] H. P. Pham, V. H. Nguyen, and N. A. Tran, "Log Analysis for Network Attack Detection Using Deep Learning Models," 2023, doi: 10.1145/3628797.3628943.
- [18] S. Kumar, V. Murgai, D. Singh and I. Kommeneni, "Recurrent Neural Network Architecture for Communication Log Analysis," *2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Kyoto, Japan, 2022, pp. 503-508, doi: 10.1109/PIMRC54779.2022.9978048.
- [19] J. Lee, Y. Lee, M. Jin, J. Kim, and J. Hong, "Analysis of application installation logs on Android systems," in *Proceedings of the 2019 International Conference on Network and Service Management (CNSM)*, 2019, pp. 2140-2145, doi: 10.1145/3297280.3297489.
- [20] J. P. Aguiar, V. Silva, V. G. O. Pontes, and P. A. da Paz Elleres, "Logcat analysis of mobile devices to identify errors," *International Journal of Advanced Research*, vol. 11, no. 11, pp. 1050-1059, 2023, doi: 10.21474/ijar01/17909.
- [21] S. Karagiannis, L. L. Ribeiro, C. Ntantogian, E. Magkos, and L. M. Campos, "Chidroid: A Mobile Android Application for Log Collection and Security Analysis in Healthcare and IoMT," *Appl. Sci.*, vol. 13, p. 3061, 2023. doi: 10.3390/app13053061.
- [22] J. Zhu, S. He, P. He, J. Liu and M. R. Lyu, "Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics," *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, Florence, Italy, 2023, pp. 355-366, doi: 10.1109/ISSRE59848.2023.00071.

- [23] M. Awais, "Successful Data Cleaning and Preprocessing for Effective Analysis," Brickclay. Accessed: Jul. 23, 2024. [Online]. Available: <https://www.brickclay.com/blog/machine-learning/successful-data-cleaning-and-preprocessing-for-effective-analysis/>
- [24] H. Li and Y. Jiang, "Improved Isolation Forest Algorithm Based on Feature Weights and Local Data Relationships," 2023 3rd International Conference on Digital Society and Intelligent Systems (DSInS), Chengdu, China, 2023, pp. 453-458, doi: 10.1109/DSInS60115.2023.10455433.
- [25] D. Deng, "Research on Anomaly Detection Method Based on DBSCAN Clustering Algorithm," 2020 5th International Conference on Information Science, Computer Technology and Transportation (ISCTT), Shenyang, China, 2020, pp. 439-442, doi: 10.1109/ISCTT51595.2020.00083.
- [26] Y. Chen, J. Qian and V. Saligrama, "A new one-class SVM for anomaly detection," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 2013, pp. 3567-3571, doi: 10.1109/ICASSP.2013.6638322.
- [27] M. Hossin, M. N. Sulaiman, A. Mustapha, N. Mustapha and R. W. Rahmat, "A hybrid evaluation metric for optimizing classifier," 2011 3rd Conference on Data Mining and Optimization (DMO), Putrajaya, Malaysia, 2011, pp. 165-170, doi: 10.1109/DMO.2011.5976522.
- [28] G. Bansal and R. Tekchandani, "Selecting a set of appropriate metrics for detecting code clones," 2014 Seventh International Conference on Contemporary Computing (IC3), Noida, India, 2014, pp. 484-488, doi: 10.1109/IC3.2014.6897221.
- [29] T. H. Dinh, C. H. Le and Q. Ha, "UAV Imaging: Correlation Between Contrast and F1-Score for Vision-Based Crack Detection," 2024 IEEE/SICE International Symposium on System Integration (SII), Ha Long, Vietnam, 2024, pp. 657-662, doi: 10.1109/SII58957.2024.10417452.
- [30] P. Adeodato and S. Melo, "A geometric proof of the equivalence between AUC_ROC and Gini index area metrics for binary classifier performance assessment," 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 2022, pp. 1-6, doi: 10.1109/IJCNN55064.2022.9892048.
- [31] S. Navyakala and R. Rathinam, "Intrusion Detection System Using Hybrid Model of Denoising Autoencoder and Ladder Variational Autoencoder," 2024 10th International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 2024, pp. 214-219, doi: 10.1109/ICCSP60870.2024.10543519.
- [32] Z. Guan, J. Wang, X. Wang, W. Xin, J. Cui and X. Jing, "A Comparative Study of RNN-based Methods for Web Malicious Code Detection," 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), Chengdu, China, 2021, pp. 769-773, doi: 10.1109/ICCCS52626.2021.9449245.
- [33] Y. Jiang, W. Wang and C. Zhao, "A Machine Vision-based Realtime Anomaly Detection Method for Industrial Products Using Deep Learning," 2019 Chinese Automation Congress (CAC), Hangzhou, China, 2019, pp. 4842-4847, doi: 10.1109/CAC48633.2019.8997079.
- [34] J. O. Loshima, "Android-Log-Anomaly-Detection: Python scripts for detecting anomalies in Android logs using machine learning and deep learning algorithms, including Isolation Forest, One-Class SVM, Autoencoders, DBSCAN, RNN, CNN, and LSTM," GitHub, 2024. [Online]. Available: <https://github.com/juniorLoshima/Android-Log-Anomaly-Detection>. [Accessed: Sep. 12, 2024].