Online Learning-Based Android Malware Detection Using API Call Graphs and Drift Detection: A Comparative Study

Mohammed Daawar Hussain[1], **Ali Muzaffar**[1]

[1]Heriot-Watt University, Dubai, UAE
School of Mathematical and Computer Sciences
mdh2000@hw.ac.uk, ali.muzaffar@hw.ac.uk

Abstract

The rapid growth and complexity of Android applications have made the platform a serious target for cybercriminals, posing substantial risks to mobile security and user data. Traditional malware detection models, although they have shown promise, can hardly be applied at run-time since they cannot adapt quickly enough to new malware variants and evolving attack methods. Such models, trained on preexisting data, suffer from performance degradation due to concept drift, where data distributions change over time as malware evolves. This paper presents an Online Learning-Based Android Malware Detection framework that systematically pairs various drift detection algorithms—such as ADWIN, DDM, and EDDM—with various machine learning models to identify the most effective combinations for maintaining detection accuracy in real-time. Our best-performing model achieved an accuracy of up to 96.01%.

Keywords: Malware detection; Android security; Machine learning; Online learning; Drift detection

Introduction

In an era where mobile technology is indispensable, the Android platform has become a dominant force, powering billions of devices worldwide. According to StatCounter (2024), Android currently commands 71.17% of the global mobile operating system market, emphasising its extensive use and broad reach. However, this widespread adoption, coupled with Android's open-source nature, makes it a frequent target for malicious actors who exploit app vulnerabilities and compromise user privacy and data security. In response, machine learning (ML) has emerged as a powerful tool for malware detection, offering the ability to automatically identify complex and evolving threat patterns that rule-based systems may miss (Muzaffar et al. 2023a).

Traditional ML classifiers such as Decision Trees, Random Forests, SVMs, and Logistic Regression have been used with both static and dynamic analysis to improve detection accuracy. However, these models are often trained offline on fixed datasets, limiting their ability to adapt in real-time to new, dynamic threats. A key challenge is concept drift, where malware behaviour changes over time, leading to degraded performance if models are not regularly updated (Liu et al. 2020).

Online learning models such as Passive Aggressive (PA) (Crammer et al. 2006), Hoeffding Trees (Domingos and Hulten 2000), Adaptive Random Forests (ARF) (Gomes et al. 2017), and Stochastic Gradient Descent (SGD) (Bottou 2010) offer incremental learning capabilities that support real-time updates. Several frameworks like DroidOL (Narayanan et al. 2016), CASANDRA (Narayanan et al. 2017), and AIBL-MVD (Darem et al. 2021) have demonstrated the practical benefits of these models, though many overlook or simplify the problem of drift. To explicitly address evolving malware behavior, concept drift detectors such as DDM (Gama et al. 2004), EDDM (Baena-García et al. 2006), ADWIN (Bifet and Gavaldà 2007), PHT (Page 1954), and Hoeffding-based methods (Frías-Blanco et al. 2015) have been proposed, yet a comprehensive evaluation across multiple detectors and classifiers remains lacking.

This study builds upon the **ActDroid** framework (Muzaffar et al. 2024a) by evaluating a range of drift detection techniques in conjunction with multiple ML models, to identify the most effective combinations for enhancing accuracy in real-time Android malware detection. It presents a comparative analysis of drift detection algorithms within an active learning setting, investigates their empirical impact on model performance, and offers practical recommendations for selecting optimal model-detector pairings. Through this, the study aims to improve the adaptability and long-term robustness of malware detection systems in dynamic threat environments.

Methodology

Dataset We use the "Android Dataset for Malware Detection" (Muzaffar et al. 2024b, 2023b), obtained from the Heriot-Watt University Research Portal, which contains 16,208 Android applications—8,106 benign and 8,102 malicious—collected between 2019 and 2021. For this study, only static API call graphs were retained for modelling, as prior research (Muzaffar et al. 2023a) found dynamic features (e.g., system calls, traffic logs) offer marginal gains relative to their computational cost. To simulate a real-time streaming environment, apps were ordered chronologically by release date; in cases where malware lacked known release dates, an estimated appearance date was calculated as

Model	DDM	EDDM	PHT	ADWIN	HDDM_A
ARF	5	18	6	6	4
HT	5	18	6	7	1
PAC	0	0	4	2	0
SGD	0	9	1	1	0

Table 1: Drift detection counts for each model.

40 days prior to their VirusTotal submission, accounting for typical threat detection delays.

Feature Selection Each app is converted into a directed call graph, where nodes represent API calls and edges denote execution order (API Call graph); acyclic graphs are topologically sorted, while cyclic ones undergo breadth-first traversal to extract a space-separated API call sequence. These sequences are then transformed into high-dimensional sparse vectors using TF-IDF, with 5–7 length n-grams and a vocabulary limited to the top 100,000 terms for scalability. To enhance model efficiency and relevance, mutual information-based filtering is applied to retain the top 30,000 most informative features for training.

Model Selection We evaluate four online learning models for their suitability in handling streaming data efficiently: Passive Aggressive Classifier (PAC), Hoeffding Tree (HT), Adaptive Random Forest (ARF), and Stochastic Gradient Descent (SGD). All models were implemented using the `River` library with consistent preprocessing and vectorized API call graph features.

Drift Detection Algorithms To address concept drift in streaming malware detection, we evaluate five commonly used drift detection algorithms: DDM, EDDM, HDDM_A, ADWIN, and PHT. These were selected because of their differing sensitivities to abrupt and gradual drifts. All detectors are used with their default configurations from the `River` library (Montiel et al. 2021).

Experimental Framework The proposed experimental framework is an Online Learning-based Android malware detection system designed to incrementally adapt to threats using real-time model updates and integrated concept drift detection. Android APKS are collected from public sources and labelled using VirusTotal results. Each app is represented through API call graphs, which are transformed into TF-IDF vectors for model input. The `River` library is used to deploy online learning models and drift detectors, enabling step-by-step predictions, feedback from ground truth labels, and real-time drift monitoring. Upon drift detection, the model is reset to prevent performance degradation due to outdated patterns. This process allows real-time learning without requiring full retraining on the entire dataset, allowing adaptation to changing malware patterns whilst maintaining lightweight resource requirements.

Results & Discussion

The experimental results show that model performance is closely tied to the interaction between each model's adaptability and the sensitivity of its paired drift detector.

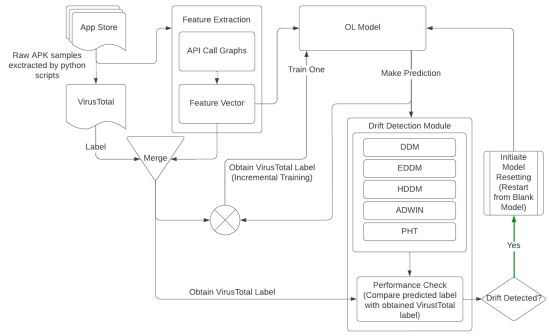

Figure 1: Experimental Online Learning Framework

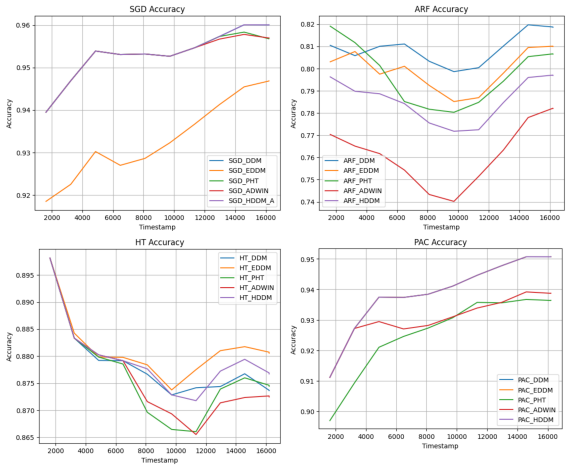

Figure 2: Accuracy over time for all the models with different drift detectors.

The PAC and SGD model achieved the highest accuracies (94–96%) when no resets occurred, showing stability in mostly stable data streams. However, without drift handling, these models may struggle with abrupt changes. Drift detectors help address this by prompting timely resets, though frequent or unnecessary resets can lower final accuracy.

Among the detectors, DDM and HDDM_A were conservative, triggering few resets and supporting stable learning, especially with PA. EDDM was more sensitive to gradual changes but sometimes overreacted, reducing effectiveness when the drift was minor. ADWIN and PHT provided a middle ground, identifying moderate drift and correcting model errors earlier, though sometimes at the cost of stability in mostly consistent data.

Model-wise, PAC and SGD benefited from continuous linear updates, while Hoeffding Trees reached 88–89% when paired with drift detectors. Adaptive Random Forests performed moderately overall but excelled in abrupt-drift scenarios when combined with balanced detectors like DDM or PHT.

In summary, no single model-detector combination was universally superior, but rather the best results depended on

Model	Accuracy
Adaptive Random Forest (ARF)	76.82%
Hoeffding Tree (HT)	87.14%
Passive-Aggressive Classifier (PA)	93.97%
Stochastic Gradient Descent (SGD)	95.38%

Table 2: Mean Accuracy Across All Models (No drift detectors incorporated)

how volatile the data stream was. This highlights the need to align drift detector sensitivity with expected concept drift patterns to maintain both accuracy and adaptability.

Conclusions

This study presented a real-time Android malware detection framework combining online learning with drift detection to adapt to evolving threats without full retraining. Among the models and detectors evaluated, the performance varied by drift type, with Passive-Aggressive and SGD achieving the highest accuracy when resets were minimal. Although effective, the framework assumes immediate label availability and has yet to be tested in extremely high data velocity streams. Future work should explore delayed labelling, memory-efficient models, and adaptive neural networks to enhance scalability and robustness, with potential applications that extend to other dynamic cybersecurity domains.

References

Baena-García, M.; del Campo-Ávila, J.; Fidalgo, R.; Bifet, A.; Gavaldà, R.; and Morales-Bueno, R. 2006. Early drift detection method. In *Proceedings of the International Workshop on Knowledge Discovery from Data Streams (IWKDDS'06)*, 77–86. Berlin, Germany: Workshop Proceedings.

Bifet, A.; and Gavaldà, R. 2007. Learning from Time-Changing Data with Adaptive Windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 443–448. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics (SIAM).

Bottou, L. 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. In Lechevallier, Y.; and Saporta, G., eds., *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, 177–187. Paris, France: Springer.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7: 551–585.

Darem, A. A.; Ghaleb, F. A.; Al-Hashmi, A. A.; Abawajy, J. H.; Alanazi, S. M.; and Al-Rezami, A. Y. 2021. An Adaptive Behavioral-Based Incremental Batch Learning Malware Variants Detection Model Using Concept Drift Detection and Sequential Deep Learning. *IEEE Access*, 9: 97180–97196.

Domingos, P.; and Hulten, G. 2000. Mining High-Speed Data Streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 71–80. ACM, Boston, MA, USA: ACM.

Frías-Blanco, I.; Campo-Ávila, J. d.; Ramos-Jiménez, G.; Morales-Bueno, R.; Ortiz-Díaz, A.; and Caballero-Mota, Y. 2015. Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3): 810–823.

Gama, J.; Medas, P.; Castillo, G.; and Rodrigues, P. 2004. Learning with drift detection. In Bazzan, A.; and Labidi, S., eds., *Advances in Artificial Intelligence – SBIA 2004*, volume 3171 of *Lecture Notes in Computer Science*, 66–112. Berlin/Heidelberg: Springer.

Gomes, H. M.; Barddal, J. P.; Enembreck, F.; and Bifet, A. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10): 1469–1495.

Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; and Liu, H. 2020. A Review of Android Malware Detection Approaches Based on Machine Learning. *IEEE Access*, 8: 124579–124607.

Montiel, J.; Halford, M.; Mastelini, S. M.; Bolmier, G.; Sourty, R.; Vaysse, R.; Zouitine, A.; Gomes, H. M.; Read, J.; Abdessalem, T.; et al. 2021. River: Machine Learning for Streaming Data in Python. *Journal of Machine Learning Research*, 22(123): 1–8.

Muzaffar, A.; Hassen, H. R.; Zantout, H.; and Lones, M. A. 2023a. Investigating Feature and Model Importance in Android Malware Detection: An Implemented Survey and Experimental Comparison of ML-Based Methods. *arXiv*, abs/2301.12778: 1–30.

Muzaffar, A.; Hassen, H. R.; Zantout, H.; and Lones, M. A. 2024a. ActDroid: An active learning framework for Android malware detection. *Preprint submitted to Elsevier*, -(-): 1–XX. Available as a preprint on arXiv.

Muzaffar, A.; Hassen, H. R.; Zantout, H.; and Lones, M. A. 2024b. Android Dataset for Malware Detection. Accessed: 2024-11-17.

Muzaffar, A.; Ragab Hassen, H.; Zantout, H.; and Lones, M. A. 2023b. Droiddissector: A static and dynamic analysis tool for android malware detection. In *International Conference on Applied CyberSecurity*, 3–9. Springer.

Narayanan, A.; Chandramohan, M.; Chen, L.; and Liu, Y. 2017. Context-Aware, Adaptive, and Scalable Android Malware Detection Through Online Learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 1: 157–175.

Narayanan, A.; Yang, L.; Chen, L.; and Liu, J. 2016. Adaptive and Scalable Android Malware Detection through Online Learning. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2484–2491. Vancouver, BC, Canada: IEEE.

Page, E. S. 1954. Continuous inspection schemes. *Biometrika*, 41(1/2): 100–115.

StatCounter. 2024. Mobile Operating System Market Share Worldwide. https://gs.statcounter.com/os-market-share/mobile/worldwide. Accessed: 2024-11-21.