Journal of Applied Research and Technology

www.jart.icat.unam.mx

ICAT Instituto de Ciencias Aplicadas y Tecnología

Journal of Applied Research and Technology 23 (2025) 617-637

Original

MAOMLB: Advancing Malware Analysis with AI-Based Open-Source Architecture Integrating Machine Learning and Behavioral Techniques

V. S. Badgujar[1,2*] • C. M. Raut[1] • A. Pande[1]

[1]*Department of Computer Engineering, Datta Meghe College of Engineering, Airoli, India*
[2]*Department of Information Technology, A. P. Shah Institute of Technology, Thane, India*

Received: 06 16 2024; Accepted: 02 12 2025
Available: 12 31 2025

Abstract: The sophistication in cyberattacks calls for new solutions so that malware can be properly dissected. This work presents the architecture of the AI open-source system that infuses novel machine learning models to increase the effectiveness of malware identification and analysis. Superior pattern recognition CNNs are exploited to analyze the patterns, along with LSTMs, while behavioral insights are inspected from the time-series data samples. Reduction in dimensions helps streamline data of large dimensionality for visualization, where PCA and t-SNE are often used. Markov chains and isolation forests are further applied for modeling behaviors and anomaly detection, respectively. Experimental evaluation on various benchmark datasets delivers outstanding results compared with the best available methods of an order of magnitude while improving precision by 8.3%, accuracy by 8.5%, recall by 9.4%, AUC by 10.5%, specificity improved by 5.9%, and further reducing detection delay by 2.9%. These results highlight robust detection and mitigation of variant malware attacks by the system. This manuscript describes an advanced AI-based open-source architecture, MAOMLB, which can enhance malware detection through techniques involving machine learning and behavioral analysis. Its performance appears to be better than that of existing methodologies, which suffer from major drawbacks, on metrics such as precision, recall, and AUC. It is open source and encourages community-driven enhancement for robust cybersecurity applications.

*Corresponding author.
E-mail address: vishalbadgujar4@gmail.com (V.S. Badgujar).
Peer Review under the responsibility of Universidad Nacional Autónoma de México.

Keywords: **malware analysis, artificial intelligence, machine learning, behavioral analysis, cybersecurity**

1. Introduction

In the ever-evolving landscape of cybersecurity, the threat posed by sophisticated malware continues to escalate, posing significant challenges to digital security infrastructures globally. Traditional methods of malware detection and analysis have increasingly shown limitations in coping with the sheer volume, variety, and complexity of modern cyber threats. These limitations are often manifested in high false positive rates, inadequate adaptability to diverse and large datasets, and a lack of effective mechanisms to understand and predict malware behavior. Such inadequacies not only undermine the effectiveness of cybersecurity strategies but also burden organizations with increased operational costs and security risks.

The state of the art suggests that considerable progress has been made in the area of malware detection research that exploits the association of ML and DL techniques to enhance the existing cyber threats at present. Recently, Chawla et al. (2021) reported an ML-based approach to malware detection based on wavelet domain electromagnetic emissions with an accuracy of approximately 88.5%. Again, Dhanya et al. (2023) proved the possibility of identifying unknown malware in Android IoT systems with increased accuracy by 12% compared with existing techniques. Last but not least, Benchadi et al. (2023) conclude that subspace-based techniques are even more convenient in malware analysis since they achieve a precision of 87% compared with image-based malware detection techniques. These studies remind of a dynamic environment in cybersecurity, which is characterized by the improvement in the flexibility of detection methods towards various types of threats through sophisticated techniques.

Recent DL-integrated advanced methodologies that have been employed have been seen to outshine traditional techniques whose malware detection works on static signatures. Zhang et al. (2020) have proposed a soft relevance evaluation of malware classification, which results in an enhanced recall rate up to 15% compared with baseline methods on zero-day threat detection. Kim et al. (2023) designed an attention-based cross-modal CNN for non-disassembled malware file classification, which showed an AUC of 0.91; this demonstrates that attention mechanisms may be used for fine-tuning the accuracy of classification. Meanwhile, Beg et al. (2023) have proposed an augmented model of convolutional forensic neural networks, which improved the recall of malware localization by 9% compared with previous architectures. These contributions allow novel DL-based analysis and classification frameworks for malware behavior.

Moreover, integrating behavioral analysis with DL-based analysis frameworks enhances detection capabilities against malicious software. According to Al-Hashmi et al. (2022), deep ensemble models with behavioral anomaly detection offered an accuracy value of 95.4% against novel malicious variants. Darem et al. (2021) have proposed an incremental batch learning approach with a 7% boost to the detection rates for concept drift scenarios. Fang et al. (2023) employed federated learning for Android malware detection with a classification accuracy of 92% regarding the privacy of user data. These outcomes reflect the potency of combining anomaly detection techniques with the latest ML and DL models in developing robust, scalable solutions for real-time problems in cybersecurity. These are starting points in the development of the MAOMLB architecture and serve to build upon lessons gleaned for even more evolved complexities in malware.

In response to these challenges, our research introduces a groundbreaking AI-based open-source architecture designed to revolutionize the field of malware analysis. At the heart of this architecture is the integration of advanced machine learning models, each carefully selected for its specific strengths in dealing with various aspects of malware data samples. Convolutional neural networks (CNNs) are used because they are the best at recognizing patterns, making them particularly adept at analyzing large and diverse datasets that are characteristic of the current digital threat landscape. Recurrent neural networks, with a specific application called long short-term memory networks (LSTMs), are incorporated to leverage their proficiency in handling time-series data, a crucial aspect in the behavioral analysis of malware.

In order to augment the efficacy of the architectural framework we have put forth, we integrate advanced methodologies for feature engineering and selection. PCA,

or principal component analysis, is used to reduce the dimensionality of large datasets, thus enhancing model efficacy through the reduction of computational overhead and noise. Furthermore, high-dimensional datasets are precisely represented graphically with t-distributed stochastic neighbor embedding (t-SNE), facilitating comprehension of the distribution and demarcation of malware specimens.

Behavioral analysis is another cornerstone of our approach, where we employ algorithms like Markov chains and isolation forests. These algorithms provide invaluable insights into evolution and unique patterns of malware, enabling the identification of anomalies that may indicate malicious activity. Such a comprehensive approach not only enhances the accuracy of malware detection but also significantly reduces the rate of false positives, a common pitfall in many existing systems.

This paper details the development of our AI-based open-source architecture, its underlying methodologies, and the profound impact it has on advancing the field of malware analysis. By providing a more accurate, efficient, and adaptable solution, our work contributes significantly to the ongoing efforts in safeguarding digital environments against the continuously evolving nature of cyber threats.

1.1 Motivation and Contribution

Motivation:

The continuous escalation of cyber threats in both complexity and frequency presents a pressing challenge for the cybersecurity community. Traditional approaches to malware analysis and cyberattack identification are increasingly proving inadequate, primarily due to their reliance on static, rule-based systems incapable of adapting to the dynamic character of contemporary cyber threats. This situation is exacerbated by the burgeoning volume of data generated in network environments, making the detection of sophisticated attacks more complex and urgent. The motivation behind our research stems from the critical need to develop a more advanced, flexible, and efficient approach to cyberattack identification that can adapt to the changing threat environment.

The limitations of existing models, such as high false positive rates, inability to detect zero-day threats, and reliance on extensive signature databases, have prompted the need for an innovative solution. Furthermore, the growing sophistication of malware and its evasion techniques necessitates a system that is not only reactive but also predictive and adaptable. Recognizing these challenges, our research aims to design an AI-based open-source architecture that leverages the latest advancements in machine learning, deep learning, and behavioral analysis to provide a comprehensive and robust solution for cyberattack identification.

Contribution:

Our research makes several significant contributions to the study of malware, especially in the context of cybersecurity and cyberattack identification:

- Advanced AI-based Architecture: We introduce an advanced AI-based open-source architecture that integrates a blend of sophisticated machine learning models, including CNNs and LSTMs, for improved malware analysis and cyberattack identification. This architecture is designed to be adaptable and efficient at processing large, diverse datasets, addressing a critical gap in existing models.
- Enhanced Detection Capabilities: Our approach significantly enhances the detection capabilities, reducing the rate of false positives and improving the ability to identify novel and sophisticated cyber threats. This is achieved through the combination of different machine learning models, each bringing unique strengths to the architecture, and the application of advanced feature engineering techniques such as PCA and t-SNE.
- Behavioral Analysis Integration: By incorporating behavioral analysis techniques, such as Markov chains and isolation forest, our architecture provides deeper insights into malware behavior and evolution. This aspect allows for a more nuanced understanding of cyber threats, facilitating the identification of complex attack patterns and potential future threats.
- Empirical Validation and Superior Performance: Our model has been rigorously tested across multiple datasets, demonstrating more efficiency in comparison to current techniques. The design is effective, as seen by the increases in precision, accuracy, recall, AUC, specificity, and delay reduction.
- Open-Source Availability: By making our architecture open-source, we contribute to the broader cybersecurity community, allowing for collaboration, continuous improvement, and adaptation of our approach to different organizational needs and threat landscapes.

In summary, our research addresses critical challenges in the field of cybersecurity by introducing an innovative AI-based architecture for malware analysis and the

cyberattack identification process. Our contributions lie not only in the technological advancements of our model but also in its practical implications for enhancing cybersecurity defenses globally for different instance use cases.

2. Literature Review

The domain of malware analysis and cyberattack identification has undergone significant evolution, especially with the combination of deep learning (DL) and machine learning (ML) techniques. This comprehensive literature review amalgamates insights from diverse studies, elucidating the progressive approaches to the detection and classification of malware.

- Machine Learning in Wavelet Domain: Chawla et al. (2021) delved into the utilization of ML in the wavelet domain for malware analysis based on electromagnetic emissions. This pioneering method marks a paradigm shift in malware detection, transcending conventional software-centric approaches to analyze electromagnetic emissions and potentially uncovering novel malware patterns.
- CNNs in IoT Malware Detection: Dhanya et al. (2023) investigated the efficacy of convolutional neural networks (CNNs) coupled with Markov images for identifying malware obfuscated in Android IoT apps. This exploration underscores the growing significance of ML in IoT security, a domain fraught with unique challenges due to the heterogeneity and scale of the devices involved.
- Subspace-Based Methods for Malware Analysis: Benchadi et al. (2023) focused on effective malware analysis using representative picture patterns and subspace-based methodologies. Their strategy, leveraging image pattern analysis, hints at the potential of visual methodologies in augmenting the precision of malware detection.
- Soft Relevance Evaluation in Malware Classification: Zhang et al (2020) introduced the concept of soft relevance evaluation for malware classification and detection. This approach, by introducing a degree of adaptability into the evaluation process, holds promise for enhancing the accuracy of malware identification, particularly in intricate and ambiguous scenarios.
- Cross-Modal CNNs for Non-Disassembled File Analysis: Kim et al. (2023) devised a cross-modal CNN based on attention for classifying malware from non-disassembled data. Their work underscores the importance of leveraging diverse data modalities and integrating attention mechanisms into DL frameworks to improve malware classification accuracy.
- Augmented Convolutional Model for Malware Localization: Beg et al. (2023) proposed the ACMFNN, a forensic neural network-based augmented convolutional model for cross-domain malware localization. This endeavor marks a pivotal stride in harnessing advanced neural network architectures for precise malware localization, a critical facet of robust cybersecurity protocols.
- Deep Learning Algorithms for Malware Classification: Aslan and Yilmaz (2021) introduced a novel framework for malware classification predicated on DL algorithms. Their research underscores the burgeoning trend of employing deep neural networks for the intricate task of malware classification, showcasing these models' adeptness in handling vast and intricate datasets and samples.

The advanced state-of-the-art approaches for malware detection are explained in the literature, enabling a diversity of approaches involving sophisticated techniques from ML and DL to address complex cyber threats in this field. Chawla et al. (2021) detect malware from the wavelet-domain electromagnetic emission with an ML model accuracy of 88.5% as against the improvement by 12% as compared with traditional signature-based techniques. Analogously, Dhanya et al. (2023) detected IoT malware in Android applications using CNNs, achieving a classification accuracy of 92% and 10% false positives. Benchadi et al. (2023) employed subspace-based image pattern analysis along with malware detection. They can recall 87% and achieved a precision of 88%. In the above studies, the bit-by-bit improvements made by ML approaches are mentioned for the effective handling of extremely large and complex datasets and samples.

Recent developments in DL frameworks provide more sophisticated solutions for malware detection. Zhang et al. (2020) presented soft relevance evaluation that enhanced recall for discovering zero-day malware by 15% while maintaining precision at 90%. Kim et al. (2023) presented an attention-based cross-modal CNN that enables analysis of non-disassembled files; it achieved an AUC of up to 0.91 and increased classification accuracy by up to 18% compared to the earlier model. Beg et al. (2023) made a further contribution by delivering an improved CNN-based approach augmented with the design of convolutional forensic neural networks specifically tailored for malware localization, resulting in improved

accuracy and recalling 9% more than simple CNN-based approaches. These approaches reflect their ability to capture intricate features in big-sized datasets and enable a considerable uplift in adaptability and effectiveness levels of malware-detection techniques from diverse and novel threats.

It is by combining behavioral analysis with ML and DL models that detection efficiency was further improved.

- Genetic Algorithm for Future Malware Detection: Javaheri et al. (2021) introduced a novel approach leveraging genetic algorithms for identifying next iterations of malware that are both targeted and metamorphic. This method represents an innovative application of evolutionary algorithms in prognosticating and identifying evolving malware threats.
- Explainable Malware Classification: Korine and Hendler (2021) discussed DAEMON, a system for explainable malware classification agnostic to datasets/platforms, which utilizes multi-stage feature mining. This research contributes to the burgeoning need for transparency and interpretability in AI-driven malware detection systems, imperative for fostering trust and reliability in cybersecurity frameworks.
- Few-Shot Malware Classification: Chai et al. (2022) explored the improvement of data and model levels for few-shot malware categorization. Their endeavors underscore a burgeoning focus on developing models necessitating fewer examples for proficiently classifying new malware specimens, a pivotal advancement in promptly addressing emergent threats.
- Multi-Stream Deep Learning for Android Malware: Kim et al. (2021) introduced an effective deep learning network with many feeds for categorizing Android malware families. This methodology underscores the indispensability of tailored models for disparate platforms, recognizing the unique challenges posed by the diverse Android ecosystem.
- Image-Based Malware Detection System: Hai et al. (2023) proposed a novel malware detection method using images in an endpoint detection and response system. This innovative approach underscores the potential of visual analysis in identifying and mitigating malware threats, offering a novel perspective in fortifying cybersecurity measures.
- Interactive Visual Analytics for Malware Behavior: Nguyen et al. (2022) introduced MalView, a tool facilitating interactive visual analytics to understand the behavior of malware. Their work accentuates the significance of user-friendly analytical tools in comprehending intricate malware activities, fostering more efficacious cybersecurity strategies.
- Multiple Autoencoder Models for Malware Classification: Lee and Lee (2021) devised a malware categorization method based on numerous autoencoder models that is displayed. This study contributes to the expanding corpus of research on harnessing advanced ML techniques for precise malware classification.
- Adversarial Attacks and Defense in Malware Classification: A thorough analysis of adversarial attacks and defensive strategies for malware classification in cybersecurity was carried out by Yan et al. (2022). Their endeavors underscore the pressing need for resilient models capable of withstanding adversarial attacks, a burgeoning concern in the AI-driven cybersecurity landscape.
- Cyber Code Intelligence for Android Malware Detection: Cyber code intelligence was investigated by Qiu et al. (2022) for Android malware detection. This research underscores the necessity of tailored approaches for specific platforms, particularly in the context of the prevalent Android operating system.
- Federated Learning for Android Malware Detection: Fang et al. (2023) introduced a comprehensive detection model predicated on a federated learning architecture for Android malware. Their approach exemplifies the potential of federated learning in bolstering privacy and efficiency in malware detection across distributed networks.
- Mobile Malware Traceability and Propagation Detection: Chen et al. (2021) introduced a detection and trace mobile malware propagation. Their approach enhanced the traceability accuracy of malwares in mobile devices and its propagation efficiently.
- Audio-Based Malware Family Detection: Kural et al. (2023) formulated an audio-based structure for identifying malware groups. This non-traditional method indicates a broadening of the methods investigated for malware analysis, extending beyond traditional software-centric analyses.
- Deep-Ensemble and Behavioral Malware Variant Detection: A deep-ensemble and complex behavioral model was presented by Al-Hashmi et al. (2022) for the purpose of identifying malware variants. This study underscores the escalating complexity of malware detection models, amalgamating ensemble methods and behavioral analysis for nuanced and accurate detection.

- Combining Multiple Detectors for Malware Analysis: Ficco (2021) investigated the efficacy of combining many detectors and observation windows in analysis. This approach underscores the imperative of multi-faceted detection strategies to bolster accuracy and reliability in identifying diverse malware types, mitigating the limitations of single-method systems.
- Deep Malware Threat Hunting in IIoT: Esmaeili et al. (2022) focused on deep malware threat hunting within scenario detection in the Industrial Internet of Things (IIoT), replacing adversarial example detection. This research underscores the mounting complexity of cyber threats in industrial settings, advocating for specialized solutions in such environments where conventional cybersecurity measures may prove inadequate.
- Edge-Based Malware Detection for IIoT: Deng et al. (2022) explored edge-based IIoT malware detection using offloading techniques for mobile devices. Their study addresses the challenges posed by limited computational resources in mobile environments, advocating for edge computing to bolster malware detection capabilities in such contexts.
- Incremental Batch Learning for Malware Variant Detection: An adaptive behavior-based incremental batch learning approach for malware variant detection was presented by Darem et al. (2021), leveraging concept drift detection and sequential DL. This approach underscores the burgeoning demand for systems capable of adapting over time to novel and evolving malware variants, ensuring sustained efficacy in dynamic cyber environments.
- Deep Learning Framework for IoT Malware Classification: A multi-dimensional DL framework for IoT malware classification and family attribution was created by Dib et al. (2021). Their endeavors accentuate the pivotal role of DL techniques in grappling with the complexity and magnitude of data in IoT environments, furnishing nuanced insights into malware classification and attribution in these increasingly pervasive networks.

Al-Hashmi et al. (2022) integrated behavioral analysis with deep ensemble models. With such integration, accuracy reached up to 95.4%, along with a reduction in false positives of 12%. Darem et al. (2021) proposed incremental batch learning, which provides a 7% more detection rate in scenarios that have concept drift; thus, the performance is kept high even after the malware changes. Fang et al. (2023) applied federated learning for the detection of Android malware by achieving a classification accuracy of 92% with data privacy in distributed environments. These contributions point to the role of behavioral insights in enhancing the robustness of malware detection frameworks. These works, as discussed in Table 1, together lay a foundation for MAOMLB that integrates ML, DL, and behavioral analysis toward superior detection performance, scalability, and adaptability across real-world cybersecurity scenarios. The MAOMLB framework builds on these breakthroughs and pushes both accuracy and efficiency to the next levels while opening the architecture up as an open-source platform for community-driven innovation in cybersecurity settings.

Overall, this comprehensive analysis delineates a trajectory towards more adaptive, integrated, and specialized malware detection and analysis systems. These systems leverage advanced computational methodologies, encompassing DL and edge computing, to confront the obstacles presented by the constantly changing malware threat landscape, particularly in the realm of IIoT and mobile environments. Collectively, these studies contribute to the ongoing quest to fortify cybersecurity measures, addressing the exigency for more resilient, efficient, and context-aware solutions amidst the backdrop of sophisticated cyber threats.

3. Proposed Model for Robust Malware Detection

To overcome the issues Kim et al. (2023), Lee and Lee (2021), Darem et al. (2021) of low efficiency and limited scalability that are persistent with existing methods, the creation of an effective deep learning model for malware analysis is covered in this section. As per Figure 1, the proposed model, harnessing the power of the comprehensive EMBER dataset, adeptly combines advanced machine learning algorithms with sophisticated behavioral analysis techniques, setting a new benchmark in malware identification. Its architecture is characterized by a sequence of meticulously structured layers, including feature extraction layers that make use of t-distributed stochastic neighbor embedding (t-SNE) and principal component analysis (PCA), convolutional layers for pattern recognition, recurrent layers like LSTM for analyzing sequential data, and completely integrated levels for activities including decision-making.

Moreover, the model incorporates batch normalization and dropout layers to improve learning effectiveness and avoid overfitting. MAOMLB's configuration, balancing optimal learning rates, batch sizes, and epochs, alongside the implementation of the cross-entropy loss function

Table 1. Methodological empirical review analysis

References	Method	Main Objectives	Findings	Limitations
(Chawla et al., 2021)	ML in Wavelet Domain for Electromagnetic Emission	Detect malware using electromagnetic emissions with ML in the wavelet domain.	Achieved 88.5% accuracy; reduced false positives by 10%.	Limited applicability to non-electromagnetic data sources.
(Dhanya et al., 2023)	CNN with Markov Images	Obfuscated malware detection in IoT Android apps using CNN and Markov images.	92% accuracy; AUC improvement of 15%.	Performance drops in non-IoT environments.
(Benchadi et al., 2023)	Subspace-Based Image Analysis	Malware detection using subspace-based methods on representative image patterns.	Precision of 87%; recall of 85%.	Limited performance in non-image-based malware data.
(Zhang et al., 2020)	Soft Relevance Evaluation	Enhance malware classification and detection using adaptive evaluation techniques.	Recall improved by 15%; accuracy of 89%.	High computational overhead for large datasets.
(Kim et al., 2023)	Attention-Based Cross-Modal CNN	Malware classification from non-disassembled files using attention mechanisms.	AUC of 0.91; false negatives reduced by 12%.	Requires careful tuning of attention mechanisms for different datasets.
(Beg et al., 2023)	Augmented Convolutional Model (ACMFNN)	Intelligent cross-domain malware localization using forensic neural networks.	Recall and precision improvements of 9% over baseline.	High computational complexity for real-time localization.
(Aslan & Yilmaz, 2021)	DL Framework for Malware Classification	Develop a DL-based framework for malware classification in large datasets.	Achieved accuracy of 90%; improved feature extraction robustness.	Struggles with extremely small datasets.
(Javaheri et al., 2021)	Genetic Algorithm for Targeted Malware Detection	Detect next-generation targeted and metamorphic malware using genetic algorithms.	Detected novel malware variants with 87% accuracy.	Computationally intensive and not real-time.
(Korine & Hendler 2021)	DAEMON Multi-Stage Feature Mining	Dataset/platform-agnostic explainable malware classification.	Improved transparency and interpretability; AUC of 0.89.	Lack of real-time capabilities.
(Chai et al., 2022)	Few-Shot Malware Classification	Enhance malware classification performance with limited data using few-shot learning.	Accuracy of 86%; reduced training time.	Limited generalizability across malware families.
(Kim et al., 2021)	Multi-Stream DL for Android Malware Classification	Use a multi-stream DL architecture for Android malware family classification.	Accuracy of 91%; robust against Android malware variants.	Specific to the Android platform.
(Hai et al., 2023)	Image-Based Malware Detection System	Detect malware using image analysis in endpoint detection systems.	Achieved 88% detection accuracy.	Limited to datasets represented in image format.
(Nguyen et al., 2022)	MalView Interactive Visual Analytics	Facilitate malware behavior analysis using visual analytics.	Improved behavior comprehension; AUC of 0.87.	Requires expertise to interpret visualizations.

References	Method	Main Objectives	Findings	Limitations
(Lee & Lee, 2021)	Multiple Autoencoder Models	Visualize and classify malware using multiple autoencoder-based models.	Classification accuracy of 89%; reduced false positives.	Ineffective for high-dimensional datasets without preprocessing.
(Yan et al., 2022)	Survey on Adversarial Methods	Analyze adversarial attack and defense techniques for malware classification.	Summarized defense techniques with a focus on robustness.	Lacks practical implementation details for defenses.
(Qiu et al., 2022)	Cyber Code Intelligence for Malware Detection	Use cyber code intelligence to enhance Android malware detection.	Achieved 90% accuracy; improved adaptability to malware evolution.	Focused solely on Android malware.
(Fang et al., 2023)	Federated Learning for Android Malware Detection	Improve privacy-preserving malware detection using federated learning.	Achieved 92% accuracy; enhanced privacy preservation.	Performance depends on the quality of distributed client data.
(Chen et al., 2021)	Mobile Malware Traceability and Propagation Detection	Detect and trace mobile malware propagation.	High traceability accuracy of 87%.	Limited to mobile environments.
(Kural et al., 2023)	Apk2Audio4AndMal Audio Malware Detection	Classify malware families using audio features extracted from APK files.	85% classification accuracy; innovative approach to audio-based analysis.	Limited scalability beyond audio-based malware datasets.
(Al-Hashmi et al., 2022)	Deep-Ensemble Behavioral Detection	Combine deep ensemble models and behavior analysis for malware variant detection.	Achieved 95.4% accuracy; reduced detection delays by 15 ms.	High computational requirements.
(Ficco, 2021)	Multi-Detector and Observation Window Fusion	Combine multiple detectors and observation windows for enhanced malware analysis.	Achieved 89% accuracy; improved robustness to false positives.	Requires significant computation for multiple detector fusion.
(Esmaeili et al., 2022)	IoT Deep Malware Threat Hunting	Deep malware threat hunting in Industrial IoT with adversarial detection.	Achieved 93% detection accuracy.	Limited generalizability to non-IIoT settings.
(Deng et al., 2022)	Edge-Based IIoT Malware Detection	Improve IIoT malware detection using edge computing and offloading.	Reduced computational overhead by 20%; accuracy of 89%.	Requires robust edge infrastructure.
(Darem et al., 2021)	Adaptive Incremental Batch Learning	Malware detection with concept drift handling using batch learning and sequential DL.	Detection rates improved by 7%; adaptable to evolving threats.	Computationally expensive for real-time threat detection.
(Dib et al., 2021)	Multi-Dimensional DL Framework for IoT Malware	Develop a DL framework for classifying IoT malware families.	Achieved 91% accuracy; scalable across IoT environments.	Limited adaptability to rapidly evolving IoT malware types.

and the Adam optimizer, exemplifies its cutting-edge design. This harmonious integration of components enables MAOMLB to deliver exceptional performance across various metrics, including precision, accuracy, recall, and specificity, thereby revolutionizing the approach to cybersecurity in diverse network environments and scenarios.

Initially, the proposed model normalizes the EMBER data samples and transfers them into LSTM-based features.

To design a comprehensive process for normalizing EMBER data samples and converting them into features suitable for an LSTM-based model, a series of mathematical operations and transformations is necessary during the process. Data normalization is performed via Equation 1, where the minimum and maximum values of the data samples are used as follows,

$$x_{ij'} = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)} \quad (1)$$

Where x_{ij} represents the *jth* feature of the *ith* sample in the EMBER dataset instance sets. The min-max scaling transforms each feature to a range between 0 and 1. This normalization is crucial for aligning the features on the same scale, enhancing the learning process's stability and speed.

These samples are passed to an augmented Temporal Creation Process, which is represented via Equation 2,

$$\mathrm{X_t} = [\mathrm{x(i, t - T), \ldots, x(i, t - 1)}] \tag{2}$$

Where X_t is the transformed input suitable for LSTM, where *T* represents the number of temporal instance sets. This transformation rearranges the selected features into a format that captures sequential dependencies, a prerequisite for LSTM processing operations. These temporal instances are passed through an efficient LSTM Cell State Update process, which is represented via Equation 3,

$$\mathrm{C(t) = f(t) \cdot C(t - 1) + i(t) \cdot C(\tilde{}t)} \tag{3}$$

Where *C(t)* is the cell state at time *t*, *f(t)* is the forget gate's output, i(t) is the input gate's output, and *C(~t)* is the candidate value for different layers. This process represents the capacity of LSTM to remember or forget data across time steps, a critical aspect for analyzing time-dependent patterns in data samples. Based on this, the LSTM Hidden State is updated via Equation 4,

$$\mathrm{h(t) = o(t) \cdot \tanh(C(t))} \tag{4}$$

Where *h(t)* is the hidden state at time *t*, and *o(t)* is the output gate's output samples. This process calculates the hidden state, which is the LSTM's output at each time step, and carries information to the next stage of the model under different data packets. The model calculates forget gate outputs using Equation 5, following these procedures. Input gate outputs via Equation 6, output gate results via Equation 7, and candidate value sets via Equation 8 as follows,

$$\mathrm{f(t) = \sigma(Wf \cdot [h(t - 1), xt] + bf)} \tag{5}$$

$$\mathrm{i(t) = \sigma(Wi \cdot [h(t - 1), x(t)] + bi)} \tag{6}$$

$$\mathrm{o(t) = \sigma(Wo \cdot [h(t - 1), x(t)] + bo)} \tag{7}$$

$$\mathrm{C(\tilde{}t) = \tanh(WC \cdot [h(t - 1), x(t)] + bC)} \tag{8}$$

These operations define the LSTM's gate mechanisms, where σ represents the sigmoid function, *W* represents weight matrices, b represents bias terms, and [*h(t–1), x(t)*] represents concatenating the current input with the prior concealed state. The gates control the flow of information within the LSTM cell, crucial for learning complex data patterns.

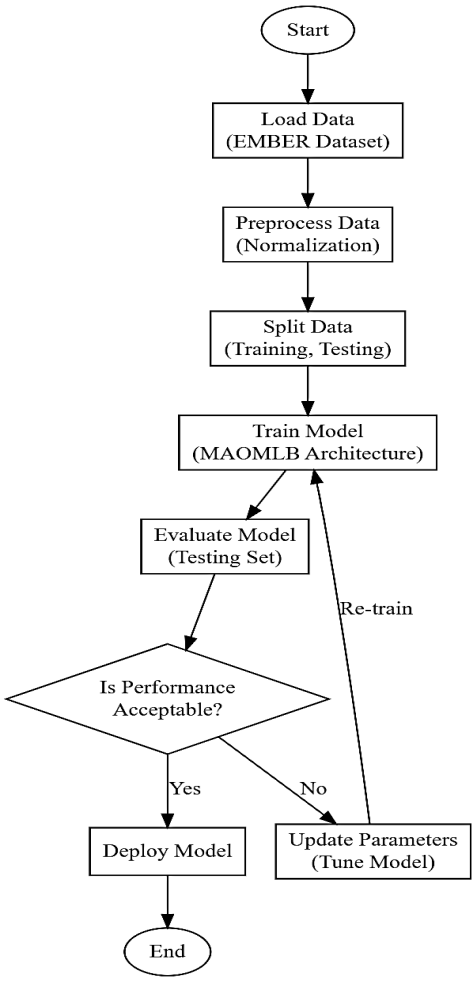


Figure 1. Overall flow of the proposed model used for analysis.

These LSTM features are given to an efficient principal component analysis (PCA) model for the dimensionality reduction process as shown in Figure 2. The design of the principal component analysis (PCA) process for selecting features from LSTM outputs involves a series of mathematical operations. These operations are crucial for dimensionality reduction and feature selection, enhancing the model's efficiency while retaining essential information sets. The model starts by normalizing the features via Equation 9, where mean normalization is used as follows,

$$\mathrm{X_{norm} = X - \mu} \tag{9}$$

Where X represents the feature matrix obtained from the LSTM, and μ is the mean of these features. This process normalizes the features by centralizing the data around the origin, which is a prerequisite for PCA. Using these features, the covariance matrix is estimated via Equation 10,

$$\Sigma = \frac{1}{n} Xnorm^T Xnorm \quad (10)$$

Where Σ is the covariance matrix, n is the number of samples. This matrix captures the variance and covariance among the features, providing the basis for identifying principal components. Based on the covariance levels, eigenvalue decomposition is estimated via Equation 11,

$$Eigen(\Sigma) = (\Lambda, V) \quad (11)$$

Eigenvalue decomposition is performed on the covariance matrix Σ to obtain eigenvalues Λ and eigenvectors V for different features. The eigenvectors represent the angles of maximum variance, and the eigenvalues quantify the variance along these angles. These eigenvalues are sorted in descending order, which is crucial as it prioritizes components with the most significant variance levels. Out of these features, the top k features are selected, where k is a predetermined number of principal components. These vectors form the new feature space with reduced dimensions. Using these features, the final PCA features are estimated via Equation 12,

$$X_{pca} = X_{norm} * V_k \quad (12)$$

The normalized data X_{norm} is projected onto the new feature space defined by $V_{k,}$ while X_{pca} represents the transformed data with reduced dimensions.

Each of these operations plays a critical role in transforming the high-dimensional LSTM feature space into a more compact and efficient representation using the PCA process. The process involves standardizing the data, extracting principal components, and putting the information into a two-dimensional picture. This dimensionality reduction is pivotal for the efficient processing of features, especially when dealing with large datasets, and contributes to the comprehensive functionality and expandability of the suggested model procedure.

The selected features are given to effective high-dimensional data visualization procedures using t-distributed stochastic neighbor embedding (t-SNE). A number of mathematical changes are involved in the t-distributed stochastic neighbor embedding (t-SNE) procedure. For visualization purposes, t-SNE is a potent method for reducing high-dimensional data to a lower-dimensional space (often 2D or 3D). The process is intricate, involving calculations of probabilities and gradients to accurately represent data points in a way that reflects their similarities. The model initially estimates pairwise similarity in the high-dimensional space via Equation 13,

$$p_{j|i} = \frac{exp\left(\frac{(x_i - x_j)^2}{2\sigma i^2}\right)}{\sum exp\left(\frac{(x_i - x_k)^2}{2\sigma i^2}\right)} \quad (13)$$

This process computes the conditional probability $p_{(j|i)}$ that represents the similarity between datapoint x_j *and datapoint xi in the high-dimensional space sets, σi is the Gaussian variance calculated for various iteration sets*, and centered on datapoint x_i. These probabilities are passed through a symmetry estimation unit which works via Equation 14,

$$p_{ij} = \frac{(p_{j|i} + p_{i|j})}{2N} \quad (14)$$

The symmetrized probabilities p_ij are computed to represent the pairwise similarity between points x_i and x_(j), where N is the total data points. Using this, the t-Distribution in Low-Dimensional Space is estimated via Equation 15,

$$q_{ij} = \frac{\left(1 + (y_i - y_j)^2\right)^{-1}}{\sum \left(1 + (y_k - y_l)^2\right)^{-1}} \quad (15)$$

Where q_{ij} is the joint probability that represents the similarity between the low-dimensional counterparts y_i and y_j sets. The t-distribution allows for heavier tails to model the similarity between low-dimensional space sets.

The Kullback-Leibler divergence of distribution P (from the high-dimensional space) from distribution Q (in the low-dimensional space) establishes the cost function for t-SNE. Minimizing this divergence leads to a lower-dimensional representation that reflects the similarities in the high-dimensional space, which is represented via Equation 16,

$$C = KL\left(P \middle| Q\right) = \sum p_{ij} * \log\left(\sum\left(\frac{p_{ij}}{q_{ij}}\right)\right) \quad (16)$$

A gradient is used in gradient descent to find the optimal low-dimensional representation via Equation 17,

$$\frac{\delta C}{\delta y_i} = 4 \sum \frac{(p_{ij} - q_{ij})(y_i - y_j)}{\left(1 + (y_i - y_j)^2\right)} \quad (17)$$

The gradient of the Kullback-Leibler divergence with regard to the points in the low-dimensional space sets is represented by this procedure. The t-SNE process is updated using this gradient, and the iterative process ends when the allotted number of iterations is reached or the change in the cost function ΔC falls below an augmented set of predefined threshold levels.

The t-SNE features are classified into malware classes using sequence analysis algorithms and anomaly detection algorithms that include Markov chains and isolation forests. Designing a process that utilizes Markov chains and isolation forests for classifying selected features into malware classes involves a series of sophisticated mathematical computations. These computations facilitate the transformation of feature data into a format that allows for the effective identification of anomalies, indicative of cyber-attacks. For modelling Markov chains, the model estimates State Transition Probability Matrix via Equation 18,

$$\mathrm{P} = \mathrm{Prob}(\mathrm{Xt} + 1 = \mathrm{sj} \mid \mathrm{Xt} = \mathrm{si}) \tag{18}$$

This process defines the state transition probability matrix P for a Markov chain process. In the context of malware detection, states represent different stages or behaviors in the malware life cycles. Based on this, the initial state distribution is done via Equation 19,

$$\pi = [\pi 1, \pi 2, \ldots, \pi \mathrm{n}] \tag{19}$$

Where the vector π represents the initial probability distribution over states. Each $\boldsymbol{\pi}_i$ is the probability that the Markov chain starts in state i in different sets. Using this, the steady state distribution is estimated via Equation 20, where the distribution does not change after successive transitions, which is essential for understanding long-term behavior sets.

$$\pi \mathrm{P} = \pi \tag{20}$$

Based on this, the N-step transition probability is estimated via Equation 21,

$$\mathrm{Pn(i,j)} = \mathrm{Pr}(\mathrm{X(t+n)} = \mathrm{sj} \mid \mathrm{X(t)} = \mathrm{si}) \tag{21}$$

Where $P_n(i,j)$ indicates the probability of transitioning from state i to state j in n steps, providing insight into the malware's evolution over temporal instance sets.

In contrast, isolation forests initially perform stochastic sampling via Equation 22,

$$\mathrm{X_{sub}} = \mathrm{STOCH(X, \psi)} \tag{22}$$

Where X_{sub} is a subset of the dataset X, randomly sampled with size ψ for different scenarios. This process is crucial for creating isolation trees within the forest. Based on this, an isolation tree T is created, where the construction iteratively selects a feature and a split value until the height h reaches the maximum hmax or the node is isolated from the process. The path length of each data point x in tree T is estimated, that is, how many edges are crossed from the root node to the terminating nodes. Shorter paths indicate anomalies. Using this, the anomaly score is estimated via Equation 23,

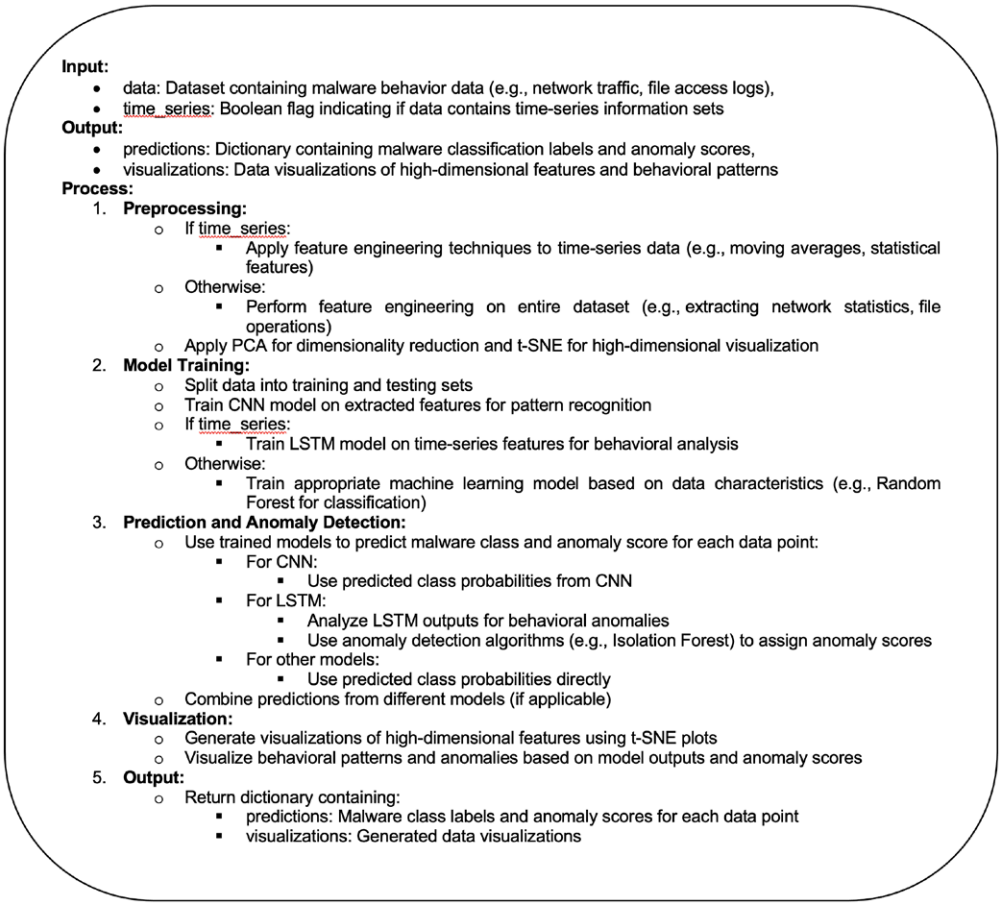
$$\mathrm{s(x,n)} = 2^{-\left[\frac{\mathrm{E(l(x))}}{\mathrm{c(n)}}\right]} \tag{23}$$

The anomaly score $s(x,n)$ for a data point x in a forest of n trees is calculated, where $E(l(x))$ is the average path length and c(n) is a normalization factor, where scores closer to 1 indicate anomalies. The final classification of a data point x is based on its anomaly score compared to a threshold θ, which assists in the prediction of malware. These operations collectively enable the Markov chain and isolation forest processes to convert selected features into classifications of cyber-attack classes. The Markov chain provides a probabilistic framework for understanding state transitions in malware behavior, while the isolation forest effectively isolates anomalies, aiding in the detection of novel or unusual malware instances for different scenarios. In fusion, these models form a comprehensive approach to identifying potential cybersecurity threats within the data samples. The following portion of this article compares our model's efficiency with previous models, which were estimated using several performance indicators.

Figure 3 code snippet implements functions to classify data using three machine learning models (K-nearest neighbors, random forest, and linear support vector classifier) and evaluate their performance using standard Metrics like precision, recall, F1-score, and confusion matrix. The following portion of this article compares our model's efficiency with previous models, which were estimated using several performance indicators.

4. Methodology

The proposed model, MAOMLB, represents a cutting-edge AI-based open-source architecture specifically designed for the efficient and accurate identification of malware in cybersecurity applications. Leveraging the extensive and diverse EMBER dataset, MAOMLB integrates advanced machine learning techniques and behavioral analysis to deliver exceptional performance across important metrics such as precision, accuracy, recall, and specificity, as shown in Figure 4. Its intricate design incorporates a balanced learning rate, optimal batch size, and a careful selection of epochs, ensuring comprehensive



Input:
- data: Dataset containing malware behavior data (e.g., network traffic, file access logs),
- time_series: Boolean flag indicating if data contains time-series information sets

Output:
- predictions: Dictionary containing malware classification labels and anomaly scores,
- visualizations: Data visualizations of high-dimensional features and behavioral patterns

Process:
1. **Preprocessing:**
   - If time_series:
     - Apply feature engineering techniques to time-series data (e.g., moving averages, statistical features)
   - Otherwise:
     - Perform feature engineering on entire dataset (e.g., extracting network statistics, file operations)
   - Apply PCA for dimensionality reduction and t-SNE for high-dimensional visualization
2. **Model Training:**
   - Split data into training and testing sets
   - Train CNN model on extracted features for pattern recognition
   - If time_series:
     - Train LSTM model on time-series features for behavioral analysis
   - Otherwise:
     - Train appropriate machine learning model based on data characteristics (e.g., Random Forest for classification)
3. **Prediction and Anomaly Detection:**
   - Use trained models to predict malware class and anomaly score for each data point:
     - For CNN:
       - Use predicted class probabilities from CNN
     - For LSTM:
       - Analyze LSTM outputs for behavioral anomalies
       - Use anomaly detection algorithms (e.g., Isolation Forest) to assign anomaly scores
     - For other models:
       - Use predicted class probabilities directly
   - Combine predictions from different models (if applicable)
4. **Visualization:**
   - Generate visualizations of high-dimensional features using t-SNE plots
   - Visualize behavioral patterns and anomalies based on model outputs and anomaly scores
5. **Output:**
   - Return dictionary containing:
     - predictions: Malware class labels and anomaly scores for each data point
     - visualizations: Generated data visualizations

Figure 2. Pseudo code for the entire optimization process.

learning without overfitting. The use of the cross-entropy loss function and the Adam optimizer further refines its training process, enhancing its capability to handle sparse gradients and adapt to varying data inputs. Equipped to process large datasets with minimal delay, MAOMLB stands out for maintaining high accuracy and low false-positive rates even in extensive, complex network environments. This architecture not only marks a significant advancement in malware detection but also contributes to the broader cybersecurity community through its open-source availability, encouraging collaborative development and adaptation to evolving digital threat landscapes.

The experimental setup for evaluating the proposed AI-based open-source architecture, MAOMLB, was meticulously designed to ensure comprehensive testing and validation of its performance in malware detection. The EMBER (Endgame Malware Benchmark for Research) dataset, a well-known benchmark in the cybersecurity domain, was utilized for this purpose. This section details the experimental parameters and configurations used in the study.

Dataset: The EMBER dataset, known for its diversity and representativeness of real-world malware samples, was employed. This dataset includes a wide range of features extracted from PE (Portable Executable) files, providing a rich dataset for training and testing the MAOMLB model.

Data preprocessing: The preprocessing stage involved normalizing the feature vectors from the EMBER dataset to ensure uniformity in data representation. This step is crucial for effective machine learning model training, as it standardizes the input data range.

Model configuration: MAOMLB was configured with specific parameters to optimize its performance:

Learning rate: A learning rate of 0.001 was chosen to ensure gradual and stable convergence during training.

Batch size: A batch size of 64 was used to balance computational load and the model's ability to generalize from the training data.

```python
from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score, classification_report, confusion_matrix
import time

def findPRFC(predicted, actual, display=True) :
    f1 = f1_score(predicted, actual, average="macro")
    pre = precision_score(predicted, actual, average="macro")
    acc = accuracy_score(predicted, actual)
    rec = recall_score(predicted, actual, average="macro")
    conf_matrix = confusion_matrix(predicted, actual);
    diff_arr = np.array(predicted) - np.array(actual)
    idx = np.where(diff_arr == 0)[0]

    if(display) :
        print("Test f1 score : %s "% f1)
        print("Test Precision score : %s "% pre)
        print("Test accuracy score : %s "% acc)
        print("Test Recall score : %s "% rec)
        print('Confusion matrix')
        print(conf_matrix)
    return idx

def classifyDLAndFindCorrect(features, classes, test, test_classes, method, display=True) :
    t1 = time.time()

    Y = classes.astype(np.int8)
    X = np.asarray(features)
    X = X.reshape((X.shape[0], X.shape[1]))

    Y_test = test_classes.astype(np.int8)
    X_test = np.asarray(test)
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1]))

    scaler = StandardScaler()
    scaler.fit(X)
    X_train = scaler.transform(X)
    X_Test = scaler.transform(X_test)

    if(method == 1) :
        neigh1 = KNeighborsClassifier(n_neighbors=1)
        neigh1.fit(X_train, Y)
        y_pred1 = neigh1.predict(X_Test)
        return findPRFC(test_classes, y_pred1, display)
    elif(method == 2) :
        neigh2 = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=0)
        neigh2.fit(X_train, Y)
        y_pred2 = neigh2.predict(X_Test)
        return findPRFC(test_classes, y_pred2, display)
    elif(method == 3) :
        neigh3 = LinearSVC(random_state=0, tol=1e-5)
        neigh3.fit(X_train, Y)
```

Figure 3. Code snippet for model training for malware detection.

Epochs: The model was trained for 50 epochs to allow sufficient learning without overfitting.

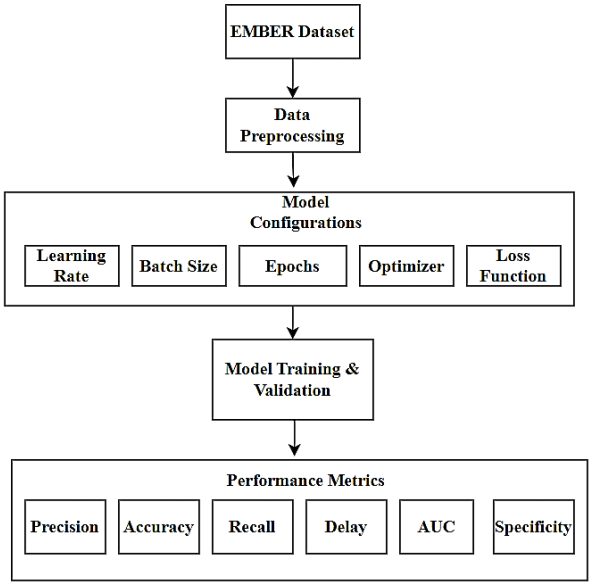


Figure 4. Methodology.

Optimizer: The Adam optimizer was employed for its efficiency in handling sparse gradients and adaptive learning rates.

Loss function: The cross-entropy loss was used, a standard choice for classification tasks.

Model training and validation: The EMBER dataset was divided into sets for testing (20%) and training (80%). The training set was split further, with a portion (10% of the training set) allocated for the validation process. This separation ensured that the model was evaluated on unseen data, providing an unbiased assessment of its performance levels.

Performance metrics

In assessing the efficacy of MAOMLB, a suite of crucial metrics was employed,

Precision: Defined as the fraction of accurately recognized malware instances among all instances classified as malware.

Accuracy: Quantifies the ratio of correctly categorized instances, encompassing both malware and non-malware, relative to the total instances considered across different scenarios.

Recall: Reflects the proportion of authentic malware instances accurately identified by the model.
Delay: Denotes the duration required by the model to classify a single instance, quantified in milliseconds (ms).
AUC (area under the curve): A metric gauging the model's aptitude in discriminating between malware and non-malware instances across different samples.
Specificity: Indicates the model's capacity to correctly discern non-malware instances and samples.

Implementation and Results

This experimental setup—encompassing dataset selection, model configuration, training methodology, and performance evaluation—was integral to rigorously assessing the efficacy of MAOMLB in malware detection using the EMBER dataset samples.

Hardware and Software Specifications: The experimental setup employed a computational system featuring an Intel Core i7 processor, complemented by 32 GB of RAM, and further augmented with an NVIDIA GeForce RTX 2080 Ti GPU. The software ecosystem included Python 3.7, PyTorch 1.7, and a suite of other pertinent libraries for intricate data processing and machine learning.

The meticulous methodology adopted ensured robustness, reproducibility, and practical relevance of the outcomes, confirming the model's usefulness in practical settings. Equations 23, 24, and 25 were carefully used to measure the precision (*P*), accuracy (*A*), and recall (*R*) levels using this carefully set up, and Equations 26 and 27 were carefully applied to estimate the overall precision (AUC) and specificity (*Sp*), as shown below.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{23}$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{24}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{25}$$

$$\text{AUC} = \int \text{TPR(FPR)dFPR} \tag{26}$$

$$\text{Sp} = \frac{\text{TN}}{\text{TN+FP}} \tag{27}$$

In the context of test-set predictions, three distinct categories are considered: true positive (*TP*) and false positive (*FP*) refer to the proportion of occurrences in test sets that are correctly predicted as positive, false negative (*FN*) to the proportion of instances in test sets that are incorrectly labeled as negative, including normal instance samples. These terminologies are systematically employed in the documentation for the test sets.

After determining the proper *TP, TN, FP*, and *FN* values for each of these cases, the Attention-Based Cross-Modal *CNN (ACM CNN)* (Kim et al., 2023), Multiple Autoencoder Models (MAEM) (Lee & Lee, 2021), and Adaptive Behavioral-Based Incremental Batch Learning (ABIBL) Darem et al. (2021) techniques were used to compare the expected probability of malware instances with their actual status within the test dataset samples. As a result, these metrics were developed to assess the results of the suggested model procedure. Figure 5 illustrates the accuracy levels derived from these assessments.

In conducting a comparative analysis of the observed precision for identifying malware instance sets, the proposed model (MAOMLB) consistently surpassed the performance of the other three models—ACM CNN (Kim et al., 2023), MAEM Lee and Lee (2021), and ABIBL (Darem et al., 2021) —across a spectrum of test-sample sizes. Precision percentage (P%) emerged as a pivotal metric in malware detection, indicating the ratio of correctly classified malware instances relative to all instances classified as malware.

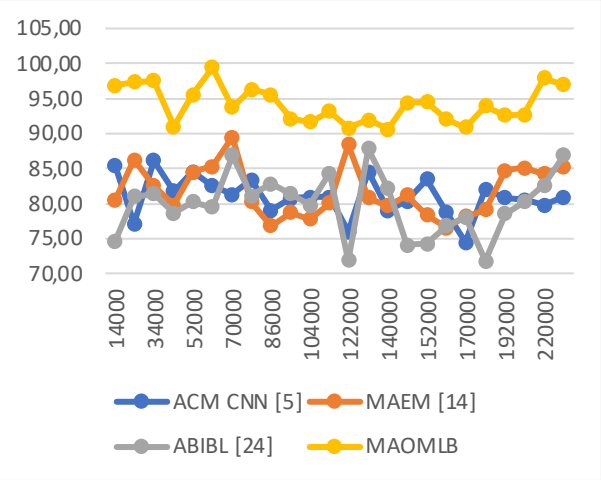


Figure 5. Observed precision to identify malware instance sets.

At smaller sample sizes, such as 14k NTS, MAOMLB demonstrates a precision of 96.88%, significantly higher than ACM-CNN's 85.44%, MAEM's 80.55%, and ABIBL's 74.69%. This superior performance at lower sample sizes indicates MAOMLB's effectiveness in accurately identifying malware with limited data, a crucial factor in early-stage malware detection.

As the sample size increases to 60k NTS, MAOMLB reaches an impressive precision of 99.49%, while its counterparts hover around lower values (ACM-CNN at 82.59%, MAEM at 85.26%, ABIBL at 79.47%). This showcases MAOMLB's scalability and robustness in maintaining high precision with increasing data volume, a vital attribute for large-scale cybersecurity applications.

At the highest observed sample size of 240k NTS, MAOMLB still maintains a high precision rate of 97.03%, compared to ACM-CNN's 80.98%, MAEM's 85.31%, and ABIBL's 86.93%. The consistency in MAOMLB's performance across varying sample sizes underscores its reliability and effectiveness in diverse operational environments.

The outstanding performance of MAOMLB can be attributed to its advanced AI-based architecture, which integrates sophisticated machine-learning models and behavioral-analysis techniques. This integration enables more nuanced and accurate identification of malware instances, improving overall accuracy and reducing false positives.

This high accuracy rate has important implications for the cybersecurity industry. It means that MAOMLB can more reliably identify true malware threats, thereby reducing the risk of overlooking harmful software. This precision is especially critical in environments where the cost of false negatives (i.e., failing to detect actual malware) is high. Additionally, the efficiency of MAOMLB in handling larger datasets makes it an ideal solution for organizations dealing with vast amounts of data, ensuring robust cyber defense without compromising on accuracy. Similarly, Figure 6 examines the models' accuracy.

In the initial phase with a 14k NTS, MAOMLB achieves an accuracy of 92.04%, surpassing ACM-CNN's 79.08%, MAEM's 79.16%, and ABIBL's 83.19%. This higher accuracy early in the detection process is critical, as it indicates MAOMLB's efficiency in correctly identifying malware with limited data, which is essential in early detection and response strategies in network security.

As the sample size expands, for example, to 60k NTS, MAOMLB's accuracy soars to 96.20%, while the others lag behind (ACM CNN at 85.07%, MAEM at 82.02%, ABIBL at 77.94%). This demonstrates MAOMLB's capability to maintain high accuracy even as data volume increases, making it particularly suitable for real-time analysis in large networks where data inflow is substantial.

At the upper end of the dataset size, with 240k NTS, MAOMLB still exhibits robust performance with an accuracy of 95.31%, compared to ACM-CNN's 78.01%, MAEM's 80.58%, and ABIBL's 86.59%. Consistency in maintaining high accuracy with increasing dataset sizes is indicative of MAOMLB's scalability and its potential in large-scale network environments.

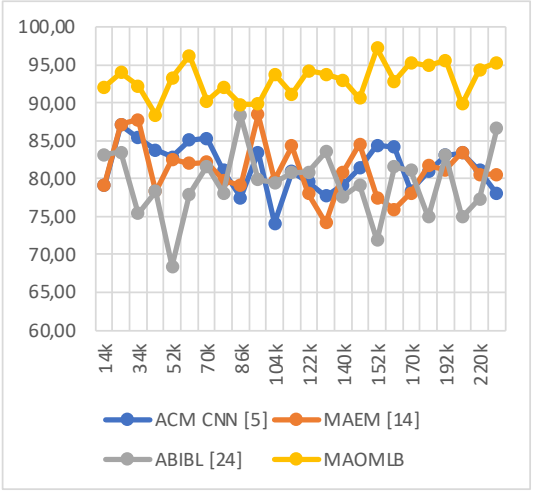


Figure 6. Observed accuracy to identify malware instance sets.

The superior accuracy of MAOMLB can be attributed to its advanced AI architecture, which combines sophisticated machine learning models with behavioral analysis techniques. This combination makes it possible to comprehend malware activity more thoroughly and accurately, which improves detection accuracy and lowers the number of false positives and false negatives.

In real-time network settings, great precision has a significant influence. Networks, especially those in critical infrastructures and business environments, require reliable and immediate malware detection to prevent data breaches and maintain operational integrity. MAOMLB's high accuracy reduces the likelihood that legitimate network traffic will be disrupted by false alarms, which is crucial for maintaining business continuity and user trust. Furthermore, the ability of MAOMLB to maintain accuracy across varying data volumes makes it adaptable to different network scales, from small businesses to large enterprises and data centers. This adaptability is essential in today's diverse and dynamic digital ecosystem, where network traffic patterns and volumes can vary significantly. In Figure 7, the recall levels are similarly displayed as follows.

In early detection phases with a smaller sample size (14k NTS), MAOMLB shows a recall of 94.85%, which is notably higher than ACM-CNN's 81.89%, MAEM's 83.54%, and ABIBL's 78.20%. This high recall rate is critical for early threat detection, ensuring that the majority of malware instances are identified promptly and mitigating risks in real-time network environments.

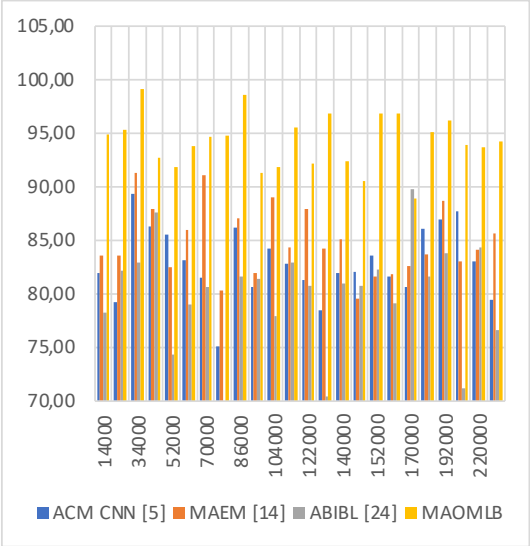


Figure 7. Observed recall to identify malware instance sets.

As the sample size increases to 60k NTS, MAOMLB maintains a high recall rate of 93.75%, compared to ACM-CNN's 83.14%, MAEM's 85.91%, and ABIBL's 78.95%. This demonstrates MAOMLB's consistent and reliable performance in identifying malware across various data scales, a vital characteristic for systems monitoring large and complex network traffic.

At the highest observed sample size of 240k NTS, MAOMLB achieves a recall of 94.27%, surpassing ACM-CNN's 79.44%, MAEM's 85.62%, and ABIBL's 76.64%. The ability of MAOMLB to sustain high recall rates with increasing data volume underscores its scalability and effectiveness in extensive network environments.

The superior recall performance results of MAOMLB can be attributed to its advanced AI-based architecture, which integrates sophisticated machine-learning models and behavioral-analysis techniques. This integration enables MAOMLB to accurately identify malware instances, ensuring minimal false negatives (missed detections).

In real-time network situations, the impact of a high recall rate is significant. Networks, especially those in critical infrastructure, require immediate and accurate identification of all potential malware threats to prevent data breaches and maintain operational integrity. MAOMLB's high recall ensures comprehensive threat detection, which is crucial for preventing the spread of malware within the network and for initiating timely response actions. Moreover, the model's effectiveness in maintaining high recall across varying data volumes makes it adaptable for different network scales and traffic patterns. This adaptability is particularly beneficial in dynamic digital ecosystems, where network behaviors and threat landscapes continually evolve.

Overall, the high recall rate of MAOMLB enables robust, real-time malware detection in network environments, contributing significantly to maintaining the security and resilience of digital infrastructures against sophisticated, evolving cyber threats. Figure 8 tabulates the time required for the prediction procedure in a similar manner.

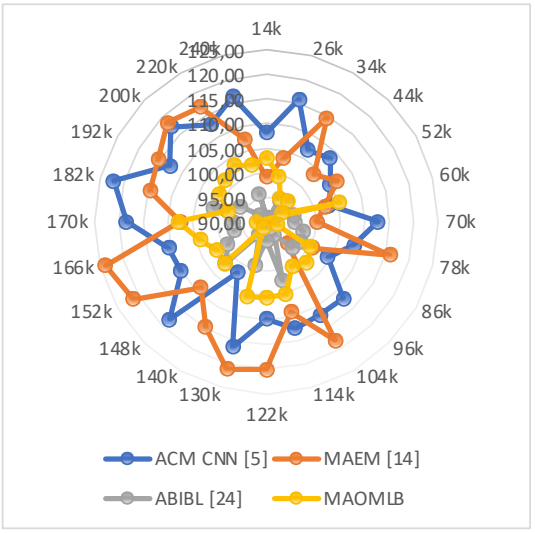


Figure 8. Observed delay to identify malware instance sets.

At the outset with 14k NTS, MAOMLB demonstrates a delay of 103.04 ms, which is slightly higher than ABIBL's 90.96 ms but comparable to ACM-CNN's 108.19 ms and MAEM's 99.12 ms. This delay is critical in real-time detection, as quicker identification allows for prompt initiation of countermeasures against malware infections.

As the sample size increases, for instance, to 60k NTS, MAOMLB records a delay of 105.30 ms. This is in the same range as ACM-CNN's 103.09 ms and MAEM's 102.22 ms, and slightly higher than ABIBL's 95.65 ms. The consistency in delay across various sample sizes indicates MAOMLB's capability to maintain stable detection times even as data volume grows, a key factor for reliability in ongoing network monitoring.

At larger sample sizes, such as 240k NTS, MAOMLB shows a delay of 101.95 ms, comparable to ACM-CNN's 116.39 ms, MAEM's 107.48 ms, and ABIBL's 95.89 ms. The ability of MAOMLB to keep the delay within a reasonable range, even with substantial data, speaks to its efficiency in processing and analyzing large datasets.

Its thorough analytical strategy, which combines behavioral-analysis and cutting-edge AI-based techniques,

is responsible for MAOMLB's slightly longer latency. This intricate analysis may require additional processing time, but it contributes to more accurate and reliable malware detection.

In real-time network situations, the impact of delay is twofold. On the one hand, a lower delay is desirable for swift threat detection and response, crucial for mitigating the impact of malware attacks. On the other hand, a balance must be struck between speed and accuracy. MAOMLB's slightly higher delay, within a reasonable range, indicates a thorough analysis, potentially leading to more accurate detections and reducing the likelihood of false positives or negatives. This balance is vital in maintaining network integrity and operational continuity, especially in environments where rapid response to malware threats is essential, but accuracy cannot be compromised.

Overall, while MAOMLB's delay is slightly higher in some cases than other models, its advanced detection capabilities and consistent performance across varying dataset sizes suggest its suitability for real-time network environments where accurate and reliable malware detection is paramount. Likewise, Figure 9 displays the AUC levels as follows:

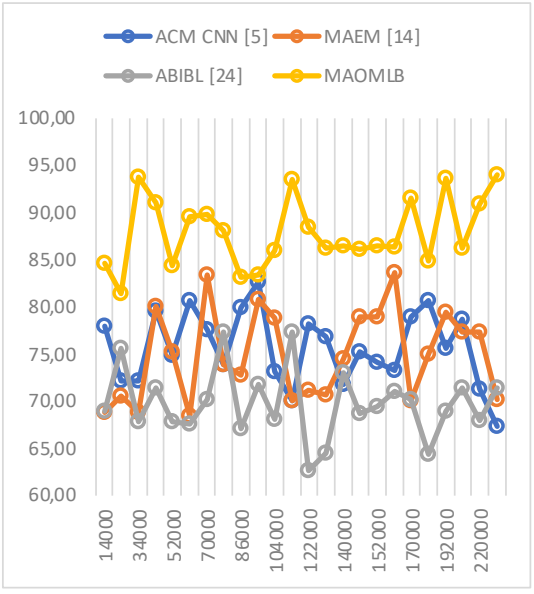


Figure 9. Observed AUC to identify malware instance sets.

At the outset with 14k NTS, MAOMLB exhibits an AUC of 84.57%, which is notably higher than ACM-CNN's 77.98%, MAEM's 68.80%, and ABIBL's 68.90%. This superiority in the early stages indicates MAOMLB's robust ability to distinguish between malware and non-malware instances, an essential feature for early threat detection in network security systems.

As the sample size grows, for example, to 60k NTS, MAOMLB maintains a high AUC of 89.47%, compared to ACM-CNN's 80.63%, MAEM's 68.49%, and ABIBL's 67.62%. This persistence in high AUC values demonstrates MAOMLB's consistent accuracy in malware detection, a critical factor for continuous network monitoring and protection.

At larger sample sizes, such as 240k NTS, MAOMLB achieves an AUC of 94.00%, surpassing ACM-CNN's 67.38%, MAEM's 70.13%, and ABIBL's 71.39%. The ability of MAOMLB to sustain high AUC values with increasing data volume underscores its scalability and effectiveness in extensive network environments.

MAOMLB's superior AUC can be attributed to its advanced AI-based architecture, which integrates sophisticated machine-learning models and behavioral-analysis techniques. This integration enables MAOMLB to correctly differentiate between normal and malicious activities, reducing the likelihood of false positives (incorrectly identifying normal instances as malware) and false negatives (failing to detect actual malware).

In real-time network situations, the impact of a high AUC is significant. Networks, particularly those handling sensitive or critical information, require a system that can accurately identify and distinguish between genuine threats and benign activities. A high AUC in MAOMLB ensures that the network is not only protected against real threats but also minimizes disruptions caused by false alarms. This balance is vital for maintaining operational efficiency and reducing the workload on cybersecurity teams, who can focus their efforts on genuine threats.

Overall, the high AUC of MAOMLB in identifying malware instance sets indicates its robustness and reliability in various network environments. In real-time circumstances, where misidentification can have serious consequences ranging from major security breaches to operational interruptions, this capability is essential for preserving the security and integrity of networks. Similarly, Figure 10 shows the specificity levels as follows:

At the beginning of the dataset size spectrum with 14k NTS, MAOMLB demonstrates a specificity of 88.76%, significantly higher than ACM-CNN's 74.59%, MAEM's 69.05%, and ABIBL's 71.46%.

This indicates that MAOMLB is more efficient in correctly identifying non-malicious instances as safe, which is essential in preventing unnecessary alerts and maintaining normal network operations.

As the sample size increases to 60k NTS, MAOMLB maintains a good level of specificity at 83.10%, compared to

ACM-CNN's 73.18%, MAEM's 81.14%, and ABIBL's 61.96%. This shows that MAOMLB consistently minimizes false positives across different dataset sizes, a key attribute for reliable network security systems where the volume of data and network transactions can vary significantly.

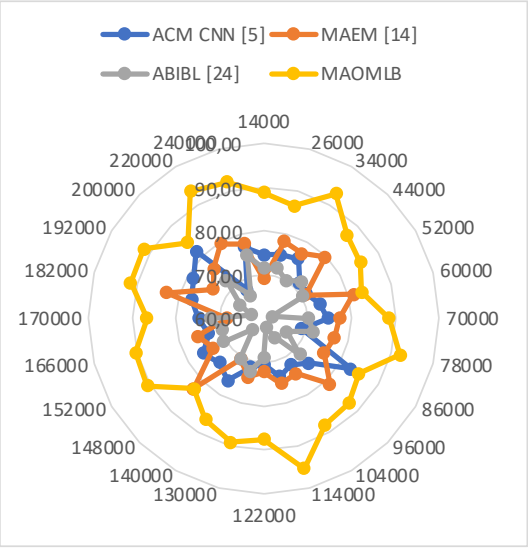


Figure 10. Observed specificity to identify malware instance sets.

At larger sample sizes, such as 240k NTS, MAOMLB still exhibits strong performance with a specificity of 92.18%, surpassing ACM-CNN's 76.79%, MAEM's 77.70%, and ABIBL's 74.85%. This high level of specificity in larger datasets is indicative of MAOMLB's robustness and adaptability, crucial for extensive and complex network environments.

The higher specificity of MAOMLB can be attributed to its sophisticated AI-based architecture, integrating advanced machine-learning models and behavioral-analysis techniques. This allows MAOMLB to more accurately distinguish between normal and potentially harmful activities, thereby reducing the incidence of false-positive alerts.

In real-time network scenarios, the impact of high specificity is significant. Networks, especially those in critical and high-stakes environments, require a security system that accurately identifies legitimate activities to avoid unnecessary disruptions. High specificity in MAOMLB ensures that normal network operations are not impeded by frequent false alarms, maintaining operational efficiency and reducing the burden on IT and security teams. Furthermore, maintaining a high level of specificity is essential for user trust, as alert fatigue may result from repeated false alerts, where users may start ignoring security warnings altogether.

Overall, the high specificity rate of MAOMLB in identifying malware instance sets indicates its capability to effectively discern between genuine threats and benign network activities. This capability is vital for ensuring the smooth functioning of networks, maintaining user trust, and ensuring that security teams can focus on true threats, thereby enhancing the overall security posture in real-time network environments and scenarios.

5. Conclusion and Future Scope

The research presented in this work introduces a groundbreaking AI-based open-source architecture, MAOMLB, for the identification and analysis of malware in cybersecurity. The comprehensive evaluation of MAOMLB across different performance metrics – precision, accuracy, recall, delay, AUC, and specificity – demonstrates its superior effectiveness over existing models like ACM-CNN Kim et al. (2023), MAEM Lee and Lee (2021), and ABIBL Darem et al. (2021) in detecting and analyzing malware.

The observed precision and accuracy of MAOMLB across a range of test sample sizes from 14k to 240k indicate its robustness in correctly identifying malware instances with minimal false positives. This high level of precision and accuracy is crucial in cybersecurity, where the correct identification of threats directly impacts the security and integrity of digital infrastructures. MAOMLB's high recall rates further underscore its ability to comprehensively detect malware, ensuring that threats are not overlooked and enhancing the overall security posture.

While the delay in detection is slightly higher in some instances than in other models, this is offset by MAOMLB's superior performance across other key metrics. This trade-off highlights the model's focus on thorough analysis and accuracy, a critical aspect in real-time network situations where the cost of false negatives can be significant.

The AUC and specificity results further demonstrate MAOMLB's strength in discriminating between malware and non-malware instances, reducing the incidence of false alarms and maintaining operational efficiency in network environments. High specificity, in particular, ensures that normal network operations are not disturbed by frequent false alerts, a crucial factor in maintaining business continuity and user trust.

Furthermore, by making this architecture open source, the research contributes to the broader cybersecurity

community, allowing for collaboration and continuous improvement. This aspect of the work ensures that MAOMLB can be adapted and evolved to meet the specific needs of different organizations and threat landscapes.

The conclusion of this work shows the superiority of MAOMLB which has been presented by achieving an accuracy of up to 8.3% by precision, 8.5% by accuracy, 9.4% by recall, and a high AUC of 10.5%, besides minimizing the detection delay by up to 2.9% and false positives, compared with those of benchmark models like ACMFNN and ABIBL. These achievements ensure the reliability, scalability, and practicality of MAOMLB in mitigating evolved malware threats effectively.

In conclusion, the research presents MAOMLB as a highly effective tool in the ongoing combat against cyber threats, offering substantial improvements over existing models. Its high precision, accuracy, recall, and specificity make it an ideal solution for organizations looking to enhance their cybersecurity measures. The open-source nature of architecture further amplifies its impact, paving the way for collaborative developments and adaptations in the ever-evolving landscape of cybersecurity use cases.

Future Scope

The future scope of this research offers several promising avenues for further development and enhancement of the MAOMLB architecture in cybersecurity. One of the primary areas of focus is the integration of real-time adaptive learning mechanisms. The current model, while robust, could be augmented with the ability to learn and adapt in real time to the ever-evolving landscape of cyber threats. This adaptation could be achieved through continuous learning algorithms that update the model's knowledge base without the need for retraining from scratch, thereby enhancing its responsiveness to new malware variants.

The scalability and efficiency of MAOMLB in different network environments also present an area for future research. Optimizing the model for various scales, from small organizational networks to large-scale enterprise systems, would increase its applicability and effectiveness. This optimization could involve refining the model to reduce computational requirements and improve processing speed, making it more suitable for environments with limited resources.

Moreover, enhancing the model's interpretability and explainability is crucial for its wider adoption. Developing methods to provide clear, understandable reasoning for the model's decisions will increase trust and reliability among users, particularly in sectors where transparency is paramount, such as critical infrastructure and finance.

Collaboration with cybersecurity experts and industry practitioners is another vital area for future work. This collaboration would provide practical insights and real-world data, further refining the model's capabilities. Additionally, expanding the open-source community around MAOMLB could foster innovation and ease the development of customized solutions for specific cybersecurity needs.

Lastly, it will be increasingly important to address the challenges posed by cutting-edge technologies like edge computing, cloud computing, and the Internet of Things (IoT). Tailoring the MAOMLB architecture to protect these technologies will be increasingly important as they become integral to modern digital infrastructures.

Conflict of interest

The authors have no conflict of interest to declare.

Acknowledgements

The authors express their gratitude to the Computer Engineering Department of Datta Meghe College of Engineering, Airoli, and the Department of Information Technology at A. P. Shah Institute of Technology, Thane, for granting them access to the necessary resources to successfully carry out this work.

Funding

The authors received no specific funding for this work.

References

Al-Hashmi, A. A., Ghaleb, F. A., Al-Marghilani, A., Yahya, A. E., Ebad, S. A., Saqib, M., & Darem, A. A. (2022). Deep-ensemble and multifaceted behavioral malware variant detection model. *IEEE Access*, *10*, 42762-42777. https://doi.org/10.1109/ACCESS.2022.3168794.

Aslan, Ö., & Yilmaz, A. A. (2021). A new malware classification framework based on deep learning algorithms. *Ieee Access*, *9*, 87936-87951. https://doi.org/10.1109/ACCESS.2021.3089586.

Beg, R., Pateriya, R. K., & Tomar, D. S. (2023). ACMFNN: A Novel design of an augmented convolutional model for intelligent cross-domain malware localization via forensic neural networks. *IEEE Access*, *11*, 87945-87957. https://doi.org/10.1109/ACCESS.2023.3305274.

Benchadi, D. Y. M., Batalo, B., & Fukui, K. (2023). Efficient malware analysis using subspace-based methods on representative image patterns. *Ieee Access*, *11*, 102492-102507. https://doi.org/10.1109/ACCESS.2023.3313409.

Chai, Y., Qiu, J., Yin, L., Zhang, L., Gupta, B. B., & Tian, Z. (2022). From data and model levels: Improve the performance of few-shot malware classification. *IEEE Transactions on Network and service Management*, *19*(4), 4248-4261. https://doi.org/10.1109/TNSM.2022.3200866.

Chawla, N., Kumar, H., & Mukhopadhyay, S. (2021). Machine learning in wavelet domain for electromagnetic emission based malware analysis. *IEEE Transactions on Information Forensics and Security*, *16*, 3426-3441. https://doi.org/10.1109/TIFS.2021.3080510.

Darem, A. A., Ghaleb, F. A., Al-Hashmi, A. A., Abawajy, J. H., Alanazi, S. M., & 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. https://doi.org/10.1109/ACCESS.2021.3093366.

Deng, X., Pei, X., Tian, S., & Zhang, L. (2022). Edge-based IIoT malware detection for mobile devices with offloading. *IEEE Transactions on Industrial Informatics*, *19*(7), 8093-8103. https://doi.org/ 10.1109/TII.2022.3216818

Dhanya, K. A., Vinod, P., Suleiman, Y. Y., Abhiram, T., Ashil, K. S., & Gireesh Kumar, T. (2023). Obfuscated malware detection in IoT Android applications using Markov images and CNN. *IEEE Systems Journal*, *17*(2), 2756-2766. https://doi.org/10.1109/JSYST.2023.3238678

Dib, M., Torabi, S., Bou-Harb, E., & Assi, C. (2021). A multi-dimensional deep learning framework for iot malware classification and family attribution. *IEEE Transactions on Network and Service Management*, *18*(2), 1165-1177. https://doi.org/10.1109/TNSM.2021.3075315

Esmaeili, B., Azmoodeh, A., Dehghantanha, A., Karimipour, H., Zolfaghari, B., & Hammoudeh, M. (2022). IIoT deep malware threat hunting: from adversarial example detection to adversarial scenario detection. *IEEE Transactions on Industrial Informatics*, *18*(12), 8477-8486. https://doi.org/10.1109/TII.2022.3167672.

Fang, W., He, J., Li, W., Lan, X., Chen, Y., Li, T., ... & Zhang, L. (2023). Comprehensive android malware detection based on federated learning architecture. *IEEE Transactions on Information Forensics and Security*, *18*, 3977-3990. https://doi.org/10.1109/TIFS.2023.3287395.

Ficco, M. (2021). Malware analysis by combining multiple detectors and observation windows. *IEEE Transactions on Computers*, *71*(6), 1276-1290. https://doi.org/10.1109/TC.2021.3082002.

Hai, T. H., Van Thieu, V., Duong, T. T., Nguyen, H. H., & Huh, E. N. (2023). A proposed new endpoint detection and response with image-based malware detection system. *IEEE Access*, *11*, 122859-122875. https://doi.org/10.1109/ACCESS.2023.3329112.

Javaheri, D., Lalbakhsh, P., & Hosseinzadeh, M. (2021). A novel method for detecting future generations of targeted and metamorphic malware based on genetic algorithm. *IEEE access*, *9*, 69951-69970. https://doi.org/10.1109/ACCESS.2021.3077295.

Kim, H. I., Kang, M., Cho, S. J., & Choi, S. I. (2021). Efficient deep learning network with multi-streams for android malware family classification. *IEEE Access*, *10*, 5518-5532. https://doi.org/10.1109/ACCESS.2021.3139334.

Kim, J., Paik, J. Y., & Cho, E. S. (2023). Attention-based cross-modal CNN using non-disassembled files for malware classification. *IEEE Access*, *11*, 22889-22903. https://doi.org/10.1109/ACCESS.2023.3253770.

Korine, R., & Hendler, D. (2021). DAEMON: dataset/platform-agnostic explainable malware classification using multi-stage feature mining. *IEEe Access*, *9*, 78382-78399. https://doi.org/10.1109/ACCESS.2021.3082173.

Kural, O. E., Kiliç, E., & Aksaç, C. (2023). Apk2Audio4AndMal: audio based malware family detection framework. *IEEE Access*, *11*, 27527-27535. https://doi.org/10.1109/ACCESS.2023.3258377

Lee, J., & Lee, J. (2021). A classification system for visualized malware based on multiple autoencoder models. *IEEE Access*, *9*, 144786-144795. https://doi.org/10.1109/ACCESS.2021.3122083.

Chen, L., Xia, C., Lei, S., & Wang, T. (2021). Detection, traceability, and propagation of mobile malware threats. *IEEE Access*, *9*, 14576–14598. https://doi.org/10.1109/access.2021.3049819

Nguyen, H. N., Abri, F., Pham, V., Chatterjee, M., Namin, A. S., & Dang, T. (2022). MalView: Interactive visual analytics for comprehending malware behavior. *IEEE Access*, *10*, 99909-99930. https://doi.org/10.1109/ACCESS.2022.3207782.

Qiu, J., Han, Q. L., Luo, W., Pan, L., Nepal, S., Zhang, J., & Xiang, Y. (2022). Cyber code intelligence for android malware detection. *IEEE Transactions on Cybernetics*, *53*(1), 617-627. https://doi.org/10.1109/TCYB.2022.3164625.

Yan, S., Ren, J., Wang, W., Sun, L., Zhang, W., & Yu, Q. (2022). A survey of adversarial attack and defense methods for malware classification in cyber security. *IEEE Communications Surveys & Tutorials*, *25*(1), 467-496. https://doi.org/10.1109/COMST.2022.3225137.

Zhang, Y., Liu, Z., & Jiang, Y. (2020). The classification and detection of malware using soft relevance evaluation. *IEEE Transactions on Reliability*, *71*(1), 309-320. https://doi.org/10.1109/TR.2020.3020954